

Dr. Dobb's Journal

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

#105 JULY 1985

\$2.95 (3.95 CANADA)

Hacking Hardware

**Build the Ultimate
Print Spooler**

Add a Real-Time Clock

**Make MSDOS
Work Like Unix**

**Turbo Pascal vs.
the Standard**



Software development isn't a mountainous task once you eliminate the high C errors.

When you can find and fix bugs at the earliest possible moment, creating software stops being such an uphill grind.

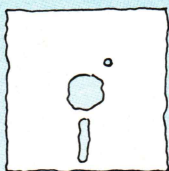
And the Smart/C Environment makes it possible. It's a complete, fully-integrated development environment for C that saves you from the creativity-inhibiting cycle of edit, compile, re-edit, re-compile, link, load, test, re-edit, re-compile, etc., ad infinitum. Smart/C puts the fun back in programming, because you spend your time creating... not waiting.

Here's why. Syntax errors are eliminated automatically as code is entered. Smart/C's highly integrated editor and interpreter allow you to interpret your program at any time in the creation process, so logic errors can be ferreted out as soon as the algorithm exists—long before any compile, link, or load.

The complete integration of the editor and interpreter means you can stop anywhere in the interpret cycle, edit, and then go right back into the interpreter exactly where you left off. Not only that, the screen-oriented user interface lets you see all operations, even interpretation, right on the listing of the code.

And to make maintenance programming easier, Smart/C's Migrator allows existing C code produced with any editor to be modified and run within the Smart/C Environment.

All of which makes Smart/C an excellent tool. It's flexible, non-restrictive, and lets you create elegant, readable, error-free programs that you can watch run with a great feeling of satisfaction.



Smart/C™

Free Demo Disk!

To fully appreciate Smart/C, you have to see it in action. For your free IBM PC MS-DOS demo disk, call us. Or write us on your company letterhead. AGS Computers, Inc., Advanced Products Division, 1139 Spruce Drive, Mountainside, NJ 07092. 800-AGS-1313. In NJ, 201-654-4321.

Smart/C Features

The Smart/C Environment

- ☐ Fully integrated editor and interpreter
- ☐ Only one load brings them both in
- ☐ One command set
- ☐ Move between one another at will

Syntax Directed Editor

- ☐ vi-like command set
- ☐ Automatically provides formats for blocks, *for*, *case* and *if* statements

Interpreter

- ☐ Current module can call external modules during interpretation
- ☐ Has Include capability
- ☐ Totally precompilation—no incremental compile
- ☐ Can interpret partially defined files allowing for rapid prototyping
- ☐ Variable speed of interpretation
- ☐ Multiple windows with user-defined sizes

The Smart/C Migrator

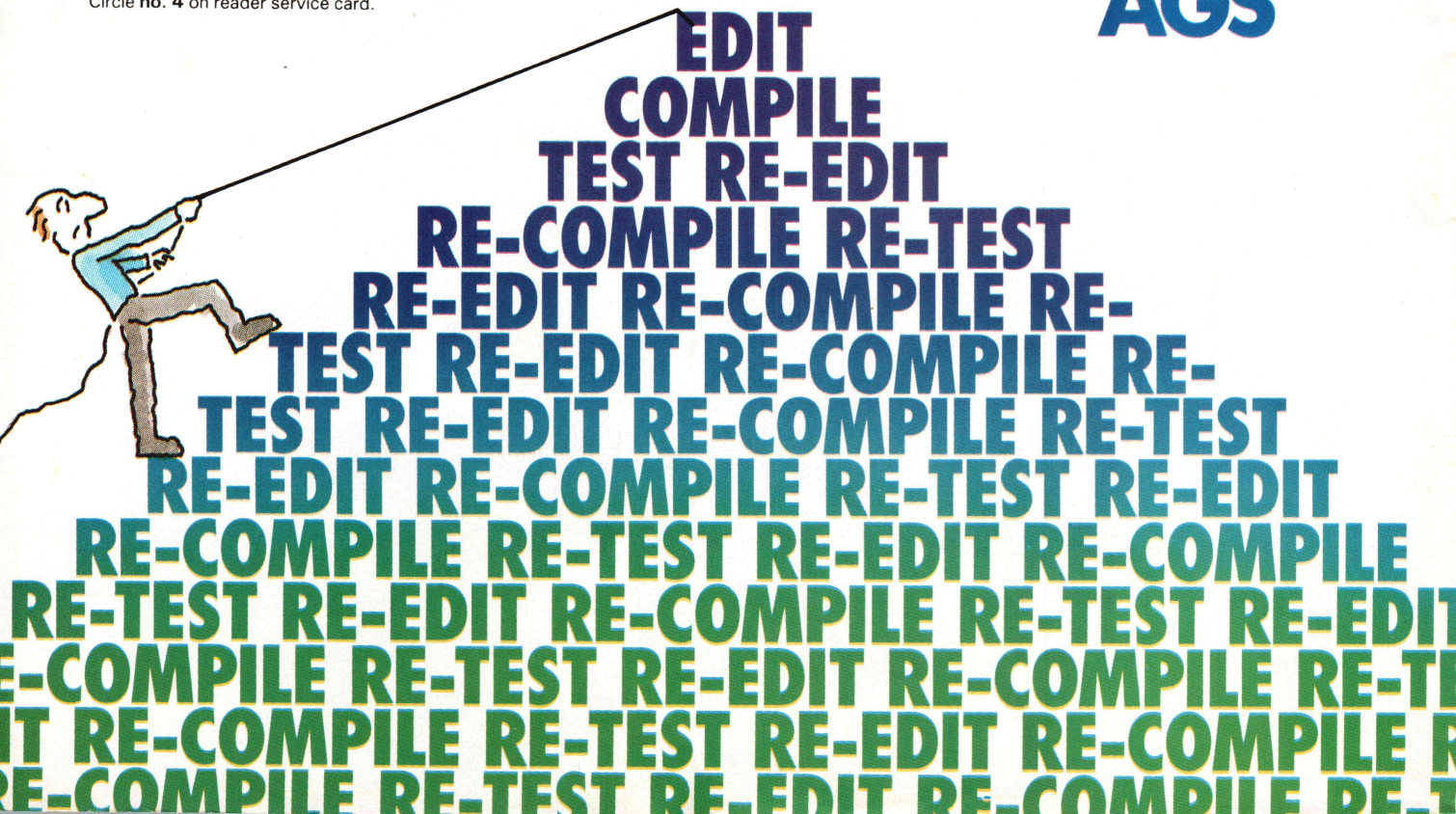
- ☐ Allows C code produced with any editor to be interpreted by Smart/C
- ☐ Reformats for readability

Smart/C has been ported to UNIX™ System V Release 2, Berkeley 4.2, Xenix,™ and MS-DOS. Versions run on 8086- and 68000-based machines, as well as proprietary architectures. Smart/C runs on PCs, micros, supermicros, minis, and even mainframes.

Trademarks—Smart/C: AGS Computers, Inc.; UNIX: AT&T Bell Labs; Xenix and MS-DOS: Microsoft Corp; IBM PC: IBM Corp.

Circle no. 4 on reader service card.

AGS



Introducing the MIX Editor

(with Split Screen - both horizontal and vertical)

A Powerful Addition To Any Programmer's Tool Box

Full Screen Editing
WordStar Key Layout
Custom Key Layouts
Terminal Configuration
Help Files
Backup Files

Introductory Offer
Only

29⁹⁵

30 Day Money Back Guarantee

Programmable
Macro Commands
Custom Setup Files
Mnemonic Command Mode
Multiple File Editing
Split Screen Editing

For PCDOS/MSDOS (2.0 and above/128K) • IBM PC/Compatibles, PC Jr., Tandy 1000/1200/2000, & others
For CPM80 2.2/3.0 (Z80 required/64K) • 8" SSSD, Kaypro 2/4, Osborne 1 SD/DD, Apple II, & others

Great For All Languages

A general purpose text processor, the MIX Editor is packed with features that make it useful with any language. It has auto indent for structured languages like Pascal or C. It has automatic line numbering for BASIC (255 character lines). It even has fill and justify for English.

Terminal Configuration

A utility for defining terminal features (smart features included) allows the editor to work with any terminal. Over 30 of the most popular terminals are built-in.

Custom Key Layouts

Commands are mapped to keys just like WordStar. If you don't like the WordStar layout, simply change it. Any key can be mapped to any command. You can also define a key to generate a string of characters, great for entering keywords.

Split Screen

You can split the screen horizontally or vertically and edit two files simultaneously.

Macro Commands

The MIX Editor allows a sequence of commands to be executed with a single keystroke. You can define a complete editing operation and perform it at the touch of a key.

Custom Setup Files

Custom keyboard layouts and macro commands can be saved in setup files. You can create a different setup file for each language you use. The editor automatically configures itself using a setup file.

Command Mode

Command mode allows any editor command to be executed by name. It is much easier to remember a command name versus a complicated key sequence. Command mode makes it easy to master the full capability of the editor. Frequently used commands can be mapped to keys. Infrequent commands can be executed by name.

Editor Commands

The editor contains more than 100 commands. With so many commands, you might think it would be difficult to use. Not so, it is actually extremely simple to use. With command mode, the power is there if you need it, but it doesn't get in your way if you don't. Following is a list of some of the commands.

Cursor Commands

Left/Right/Up/Down
Tab Right/Tab Left
Forward Word/Backward Word
Beginning of Line/End of Line
Scroll Up/Scroll Down
Window Up/Window Down
Scroll Left/Scroll Right
Top of File/Bottom of File
• • •

Block Commands

Copy/Move/Delete
Read/Write
Lower Case/Upper Case
Fill/Justify
Print

File Commands

Directory (with wild cards)
Show File/Help File
Input/Output File
Delete File/Save File

Other Commands

Split Screen/Other Window
Find String/Replace String
Replace Global/Query Replace
Delete Line/Undelete Line
Delete Word/Undelete Word
Insert Mode/Overwrite Mode
Open Line/Join Line
Duplicate Line/Center Line
Set Tab/Clear Tab
• • •

MIX 2116 E. Arapaho
Suite 363
Richardson, Tx 75081
software (214) 783-6001

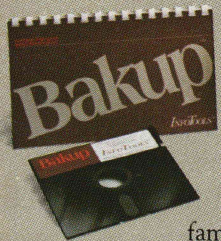
MSDOS is a trademark of Microsoft
PCDOS is a trademark of IBM
CPM80 is a trademark of Digital Research
WordStar is a trademark of MicroPro

To Order: Call Toll Free 1-800-523-9520, (Texas only 1-800-622-4070)
MIX Editor ____ \$29.95 + shipping (\$5 USA/\$10 Foreign) Texas residents add 6% sales tax

Visa ____ MasterCard ____ Card # ____ Exp. Date ____
COD ____ Check ____ Money Order ____ Disk Format ____
Computer ____ Operating System: MSDOS ____ PCDOS ____ CPM80 ____
Name ____
Street ____
City/State/Zip ____
Country ____
Phone ____

MIX 2116 E. Arapaho
Suite 363
Richardson, Tx 75081
software
Dealer Inquiries Welcome
Call (214) 783-6001
DD

TeamMate's Storage Solution: Two Minute Backup and Versatility



Two minutes to back up data?

It's true with TeamMate's

family of storage subsystems featuring the remarkable Kodak™ 2.78 megabyte flexible drive and **free Backup™** software.

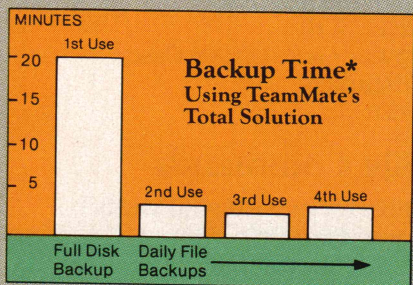
Frankly, most people know they need to protect their valuable data by backing up files regularly. They don't because it takes too long.

TeamMate has taken care of that.

All you do is back up your full hard disk once. 10 megabytes can be copied in about 20 minutes onto 3 or 4 Kodak high capacity diskettes.

Then TeamMate's easy-to-use Backup software allows you to automatically back up your modified files in as little as 2 minutes!

New advances in backup are just part of the TeamMate/Kodak success story.



TeamMate's Winchester and Kodak 2.78MB flexible drive subsystems also add unprecedented flexibility and versatility to the IBM™ PC, AT, or compatible computers.

Kodak Drives... Unlimited Storage at a Low Cost

Install them internally or externally. Either way, TeamMate's family of Kodak 2.78MB flexible drives give you Winchester-like capacity, plus the convenient familiarity of a floppy. You get eight times the online storage capacity of a standard floppy... and unlimited expandability.

The Kodak 2.78MB drive guarantees media interchangeability from drive to drive, and it can also read standard IBM PC and AT diskettes—ideal for

transporting your AT's data to standard PC's.

Need to achieve high capacity storage and rugged portability? Use our TeamMate 1103 (single Kodak drive) or the 1104, a Kodak drive and standard floppy combination—perfect for upgrading the XT. Or, use our external models featuring one or two Kodak drives to backup several PC's.

Winchester/Kodak Combinations...More Versatility

TeamMate's combined Winchester and Kodak subsystems deliver the greatest versatility available today. Use the high speed Winchester, and back up your data with the Kodak flexible drive. Also use the Kodak for data security, diskette savings, or data archiving.

Installation takes only minutes...and there's no need for software drivers or patches.

The TeamMate 1123 installs internally and adds 20 megabytes of hard disk storage plus the Kodak flexible drive. External Winchester/Kodak combinations turn a minimum PC into a sophisticated supersystem.

Save...With TeamMate Winchester Subsystems

High performance and a variety of capacities. That's TeamMate's family of Winchester subsystems.

Add 10 or 20 megabytes of storage to your system either internally or externally. For really big jobs, look to TeamMate's 1132...32 megabytes of formatted add-in storage.

Enhance a basic IBM PC AT configuration with 20 or 32 megabytes of reliable Winchester storage. TeamMate 1120 AT & 1132 AT attach directly to the AT's existing controller. Complete with drive, cable, mounting slides, and Backup software, they're ready to go in just minutes.

Adding a TeamMate to your system is the cost-effective way to handle a variety of tasks.

Call TeamMate today for more information.

Be sure to ask us about our Apple family.



Only TeamMate's Kodak Drive Solves All Your System Needs	
✓ Hard Disk Backup	High speed full or selective file backup onto high capacity diskettes. 10MB hard disk needs only 4 Kodak diskettes.
✓ Primary Storage	2.78 megabytes of online random access storage
✓ Portability	Floppy ruggedness...ideal for portable computers
✓ Expandability	Unlimited storage capacity—just change diskettes
✓ Data Security	Removeability gives you ultimate data security without security software
✓ Interchangeability	Media interchangeability guaranteed from drive to drive
✓ Read Standard Diskettes	Reads standard diskettes so you can exchange data between IBM PCs and ATs
✓ Diskette Savings	One Kodak 2.78MB diskette replaces 8 standard IBM floppies.
✓ Convenience	Install in minutes using standard DOS software



TeamMate
by Data Technology
2525 Walsh Avenue
Santa Clara, CA 95051
Tel: (408) 986-9545
Telex: 4940763 DTCSC

™ Kodak is a registered trademark of Eastman Kodak Corporation.

™ IBM is a registered trademark of International Business Machines Corporation.

™ Backup is a registered trademark of InfoTools Corporation.

Slash Programming Time in Half!

With **FirstTimeTM**

- Fast program entry through single keystroke statement generators.
- Fast editing through syntax oriented cursor movements.
- Dramatically reduced debugging time through immediate syntax checking.
- Fast development through unique programmer oriented features.
- Automatic program formatter.

FirstTime is a true syntax directed editor.

FirstTime ensures the integrity of your programs by performing all editing tasks like moves, inserts and deletes along the syntactic elements of a program. For example, when you move an IF statement, FirstTime will move the corresponding THEN and ELSE clauses with it.

Even FirstTime's cursor movements are by syntax elements instead of characters. The cursor automatically skips over blank spaces and required keywords and goes directly to the next editable position.

FirstTime is a Syntax Checker

FirstTime checks the syntax of your program statements, and also:

- Semantics like undefined variables and mismatched statement types.
- The contents of include files and macro expansions.
- Statements for errors as they are entered and warns you immediately.

FirstTime is a Program Formatter

FirstTime automatically indents statements as they are entered, saving you from having to track indentation levels and count spaces.

FirstTime has Unique Features

No other editor offer these features:

The *Zoom command* gives you a top down view of your program logic.

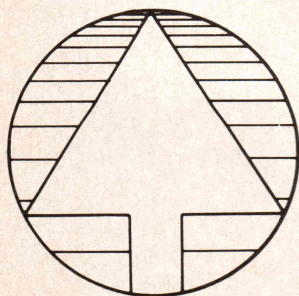
The *View command* displays the contents of include files and macro expansions. This is valuable to sophisticated programmers writing complex code or to those updating unfamiliar programs.

FirstTime's *Transform command* lets you change a statement to another similar one with just two keystrokes. For example, you can instantly transform a FOR statement into a WHILE statement.

The *Move at Same Level command* moves the cursor up or down to the next statement at the same indentation level. This is very useful. For example, you can use it to locate the ELSE clause that corresponds to a given THEN clause or to traverse a program one procedure at a time.

FirstTime is Unparalleled

FirstTime is the most advanced syntax directed editor available. It makes programming faster, easier and more fun.



TO ORDER CALL (201) 741-8188

or write:

Spruce Technology Corporation

189 E. Bergen Place
Red Bank, NJ 07701

Circle no. 65 on reader service card.

In Germany, Austria and Switzerland contact:
Markt & Technik Software Verlag
Munchen, W. Germany
(089) 4613-0

Dr. Dobb's Journal

Editorial

Editor-in-Chief Michael Swaine
Editor Randy Sutherland
Managing Editor Frank DeRose
Technical Editor Alex Ragen
Editorial Assistant Sara Noah
Contributing Editors Robert Blum,
Dave Cortesi,
Ray Duncan,
Allen Holub
Copy Editor Polly Koch
Typesetter Jean Aring

Production

Design/Production
Director Detta Penna
Art Director Shelley Rae Doeden
Production Assistant Alida Hinton
Cover Cartoon Rand Renfroe

Circulation

Sub. Fulfillment Mgr. Stephanie Barber
Subscription Mgr. Maureen Snee
Book Marketing Mgr. Jane Sharninghouse
Single Copy Sales Mgr. Kathleen Boyd

Administration

Finance Manager Sandra Dunie
Business Manager Betty Trickett
Accounts Payable Supv. Mayda Lopez-Quintana
Accounts Payable Asst. Denise Giannini
Billing Coordinator Laura Di Lazzaro

Advertising

Advertising Director

Stephen Friedman (415) 424-0600

Marketing Manager

Shawn Horst (415) 424-0600

Advertising Sales

Walter Andrzejewski (617) 567-8361

Lisa Boudreau (415) 424-0600

Beth Dudas (714) 643-9439

Michele Beaty (317) 875-0557

DDJ Classifieds

Tim Ortiz (415) 424-0600

Advertising Coordinators

Jay Horvath (415) 424-0600

Alice Abrams (415) 424-0600

M&T Publishing, Inc.

Chairman of the Board Otmar Weber
Director C.F. von Quadt
President Laird Foshay

Entire contents copyright © 1985 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0278-6508**

Subscription Rates: \$25 per year within the United States, \$46 for airmail to Canada, \$62 for airmail to other countries. Foreign subscriptions must be pre-paid in U.S. Dollars, drawn on a U.S. Bank. For subscription problems, call: outside of CA 800-321-3333; within CA 619-485-9623 or 566-6947.

Foreign Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 686-1520 TELEX: 620430 (WUI)

People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.

July 1985
Volume 10, Issue 7

CONTENTS

Referees

Michael Enright
Robert Tripp

Cover

The cover cartoon by Rand Renfroe was colored on a Via Video Graphics System by Doyle Puppo and Associates.

Editor's Note

In C chest this month Allen Holub offers an MSDOS version of the Unix directory utility ls. As the magazine went into production, we discovered that the program did not operate exactly as promised. The command ls /, which was supposed to list the contents of the root directory from another directory, did not function properly under DOS 2.x. Also, when run on a Compaq, ls did not underline directories, but printed them in half intensity. This is probably due to the particular ANSI.SYS file used.—Ed.

Dr. Dobb's Journal

ARTICLES

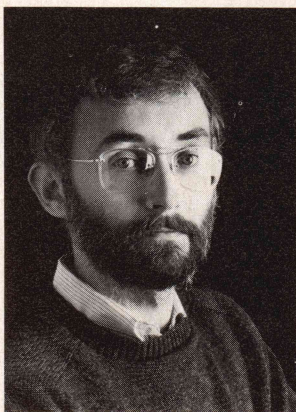
- | | | |
|---|-----------|--|
| Build a Custom PC or Clone
<i>by Jim Kronman</i> | 42 | From individual components you can assemble a PC XT work-alike that is totally IBM compatible and costs substantially less. (Reader Ballot No. 192). |
| The Ultimate Parallel Print Spooler
<i>by Don Rindsberg</i> | 46 | Build a parallel print spooler with a 60K buffer, software refresh of RAM and a peek-a-boo ROM. (Reader Ballot No. 193). |
| Designing a Real-Time Clock for the S-100 Bus
<i>by Alan Wilcox</i> | 56 | Construct a clock board for your S-100 system that prints the time, date and filename as a header on your programs. (Reader Ballot No. 194). |

COLUMNS

- | | | |
|--|------------|---|
| Dr. Dobb's Clinic
<i>by D. E. Cortesi</i> | 12 | A design error in the 8-bit version of Turbo Pascal; Turbo compared to Standard Pascal. (Reader Ballot No. 190). |
| C Chest
<i>by Allen Holub</i> | 20 | Ls, a Unix-like MSDOS directory utility; multi-column printing; a new MSDOS access function for Lattice C. (Reader Ballot No. 191). |
| Realizable Fantasies
<i>by Michael Swaine and Bob Albrecht</i> | 92 | Progress reports on fantasies presented to date. (Reader Ballot No. 195). |
| 16-Bit Software Toolbox
<i>by Ray Duncan</i> | 94 | MSDOS installable device drivers. (Reader Ballot No. 196). |
| Computer Calisthenics
<i>by Michael Wiesenber</i> | 110 | Answers to puzzles presented in previous columns. (Reader Service No. 197). |

DEPARTMENTS

Editorial	6
Letters	8
DDJ Classified	115
Of Interest	120
Advertiser Index	128



We do this for you.

"We:" that's us, the editorial staff; "this" is the magazine. It's that "you" that makes things tricky.

"You" are a statistical construct. You are 40,000 readers with at least 40,000 opinions on any random subject, 40,000 ideas about what *Dr. Dobb's Journal* should be.

You are a systems programmer for IBM. A DoD Ada specialist. You are the DP Manager for a large corporation. You are the only programmer for a small firm. You are an independent software developer. A consultant. You're a structural engineer who happens to write software. A researcher. A teacher. A student.

You program in a high-level language because you're a high-level thinker, and you use assembly language only to tighten the bolts. You have concluded that all serious computer languages are fundamentally equivalent, so why doesn't everyone just use Pascal? You think that computer languages differ widely in power, efficiency and naturalness, so why doesn't everyone use Forth (since, as surely everyone knows, Forth code is invariably faster and tighter and more readable than anything else)? You've forgotten in the past year that there is any high-level language but C.

You refuse to program in a high-level language, pouring all your creative efforts into high-torque 68K assembly language code. 8086/8088 code. Z80/8080 code. You know the 6502 inside out. You know, even if you are in the vanishing minority, that the 6809 is all the processor anyone really needs. You know things about the 80286 that they haven't discovered at Intel yet.

You know what you like about *DDJ*. The *DDJ* articles for 1985 that you were most interested in (so far) included pieces dealing with the Unix, CP/M and MSDOS operating systems; the C and Prolog languages; the IBM PC, the PC/AT, the Mac, the Commodore 64, Z80 machines and "machines in the 68000/16000 class with virtual memory."

You're intelligent, knowledgeable, creative and opinionated, but let's face it: you are not particularly consistent. Neither, consequently, are we.

This is the hardware issue of *Dr. Dobb's Journal* of "software tools for advanced programmers." Why are we publishing hardware construction articles in a software magazine?

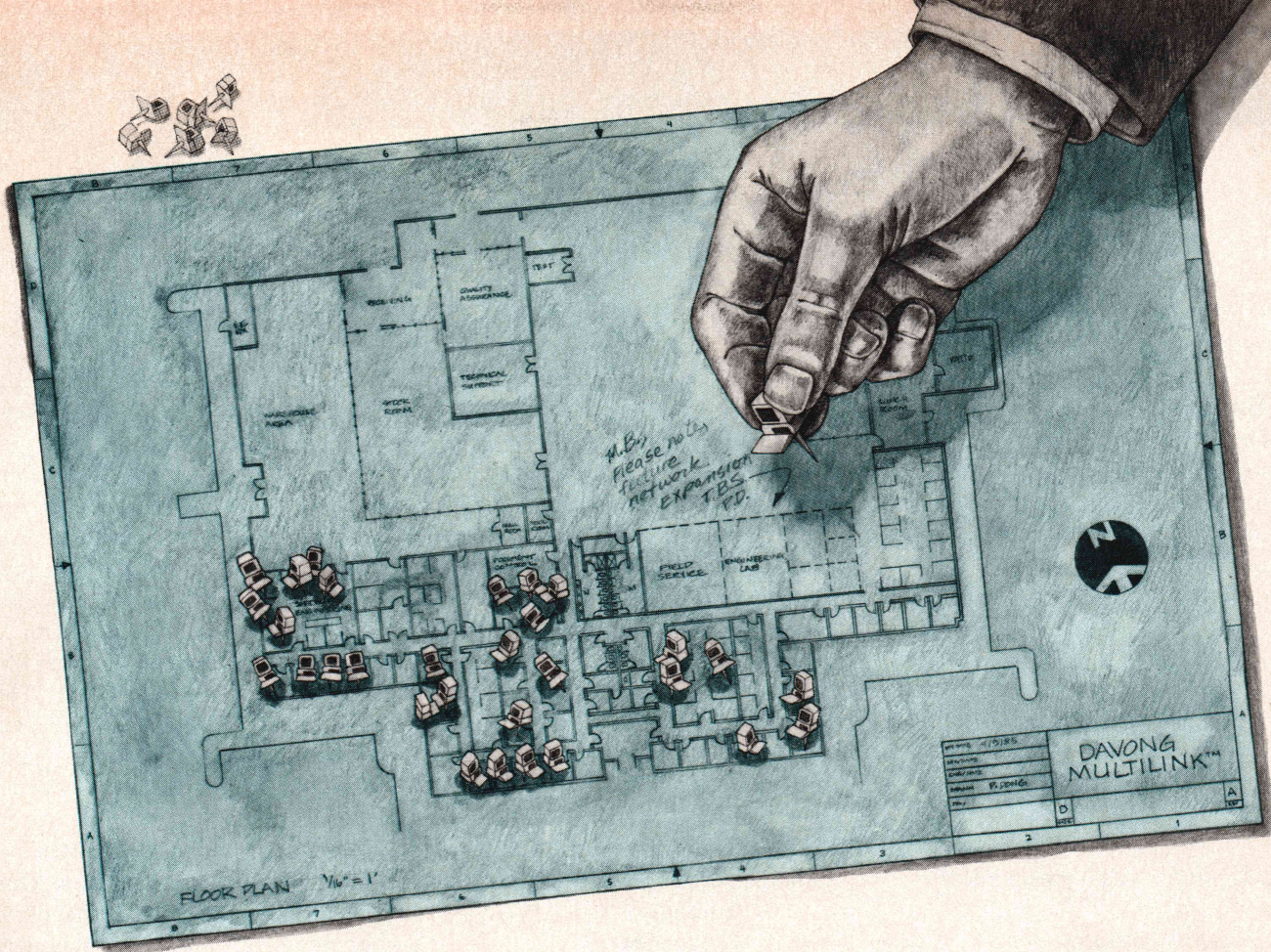
We are doing it, of course, for you. The "Fatten Your Mac" article in January was one of our most popular articles to date, and other hardware-related pieces have been well-received.

We have a broad constituency, trying as we do to address all advanced programmers, and magazine space is heartbreakingly limited. So we try to be as broad as possible without sacrificing depth. We publish generalizable code whenever we can, emphasizing good algorithms.

But the canvas tears when stretched to include hardware articles. Although there is much non-hardware material in this issue, we have this month committed a sizeable portion of the magazine to hardware-related material. We offer this collection of hardware articles with a question: how would you like to see us cover hardware in the future? With an occasional article when it seems appropriate? With an annual hardware issue like this? Or a hardware issue of a different sort? In separate monographs/project papers, leaving the magazine pages for software only? Or not at all?

Michael Swaine

Michael Swaine



Pinpointing Your Network Solution

Let's get straight to the point! If you want a powerful network to link up to 255 IBM® and IBM-compatible computers, then Davong MultiLink™ is clearly the engineered choice for you.

Let's pin down a few facts.



You don't have to be a networking wizard with MultiLink. Menus and help screens make it easy for you to control the network—not vice versa. Managing the space on one or 255 hard disks is simple.



You have worry-free expand-ability with Davong MultiLink. Nobody has to stop work just because you want to add another workstation or file server to your MultiLink network—even if you add a Davong DataSystem™ to make a file server of your PC. And, file servers also perform double-duty as workstations.



You don't have to worry about security. MultiLink provides password protection and three levels of access for each volume. There's even password protection for each hard disk! In addition, you can share volumes without collision worries. The shared DOS feature takes care of that. And, automatic file-locking capability and optional record locking. Finally! A network that provides true file-sharing!



You can have a unique boot configuration—directly from the network. No need for boot diskettes. Use space from any hard disk on the network. Keep all your files in one volume or spread them among several. MultiLink allows access to 6 active volumes at a time. And, accessing other hard disk volumes is easy.



Included with Davong MultiLink is one of the most complete print spooling packages on the market today. Print to any serial or parallel printer on the network—even multiple copies. It's easy with MultiLink.

Interested? Find out more about Davong MultiLink by contacting your local Davong dealer.

Why be stuck with less?

IBM® is a registered trademark of International Business Machines, Inc. Davong MultiLink™ and Davong DataSystem™ are trademarks of Davong Systems, Inc.

Circle no. 2 on reader service card.



Davong Systems, Inc.
217 Humboldt Court
Sunnyvale, CA. 94089
Phone: (408) 734-4900
Telex: 176386



Small C

Dear DDJ:

I don't know if anybody's still keeping track of these things, but I've found a Small C bug. The source

```
int lptr; /* global integer */
...
routine( ) {
    int r;
    char bytes[2];
    ...
    lptr += r - bytes;
    ...
```

compiled to (Z80 mnemonics)

```
... (usual stuff)
LD HL, (LPTR)
PUSH HL;;; (save left side of
              += statement)
LD HL, 4
ADD LH, SP
CALL CCGINT
EX DE, HL;;
LD HL, 2
ADD HL, SP
CALL CCSUB
POP DE      (retrieve left side of
              += statement)
ADD HL, DE  (add it)
LD A, L     (forget what's going
              on ...)
LD (DE), A  (should be LD
              (LPTR), HL)
...
```

which is entirely the wrong idea and has very nasty effects (the bug also occurs with local destinations for the assignment, but with somewhat less dangerous results). The problem disappeared when I removed the last code line from `plunge2()` (aka `plnge2` in "The Small-C Handbook" by James E. Hendrix) in `cc31.c`:

```
if ((oper == ffsb) )
```

```
(oper == ffsb))
result(lval, lval2);
```

In my copy of the compiler, this is the only place `result()` is used, so I removed it from `cc33.c`. The compiler compiled itself in this shape, and successfully compiled a reasonably large program, so maybe this is the correct fix, but I'm not deeply convinced.

Fixing the compiler is a real adventure for me, because I don't know how it works. I do wonder what solemn oath bound the original authors, so that no trace of comment is allowed to stain the pure pages of code . . .

Cordially,
Gregor Owen
μ Software/Hardware
35 Admiral St.
Port Jefferson Station,
New York, NY 11776

Other fixes and a status update for Small C will appear in the August issue—Ed.

Iconoclasm

Dear DDJ:

By your response to Frank Gaude's criticism of icons, it appears as though you don't fully appreciate his point. In fact, your apparent praise of 'active, growing icons' only strengthens his argument. What will we have if software developers everywhere start creating systems with user-definable icons? We will have the second stage in the evolution of language, as icons were the first. What is more, we will have multitudinous languages, with each programmer or user designing the icon interface to [his] own personal taste. Nobody will be able to sit down at somebody else's system and run it. We will have the Tower of Babel all over again. Confusion. Insanity. Let's not do it all over again.

For those who insist upon perform-

ing all operations with a single key-stroke, there are menus.

Another thing: metaphors. This isn't really related to icons, except that it's tangled up in your discussion of them. I wouldn't quibble with anyone who said that the purpose of software was to entertain, educate, or to simply get the job done . . . or to obfuscate, mislead, and lie. But 'The purpose of software is to realize metaphors'? Could you say that again, please? Could you illustrate with an example?

Well, gee, now that I've gotten going it's just too hard to stop. One more thing. I greatly doubt that all of your readers know that Richard Conn wrote ZCPR3. As a consequence, I would have thought it to be a prudent editorial decision to add a note to that effect to his review [DDJ #103, May 1985] of the Ampro computers, which comes bundled with ZCPR3. What do you think?

(By the by, I mean no criticism of ZCPR3. It's the best damn thing to come down the pike since the Z80. Conn—and Echelon—are to be applauded roundly.)

Sincerely,
Dreas Nielsen
234 NW 30th St.
Corvallis, OR 97330

Indeed we should have identified Richard Conn's affiliation.—Ed.

CP/M

Dear DDJ:

I have been using CP/M 2.2 for several years, and acquired CP/M Plus about 9 months ago. I am pleased with it, but I have run up against one difficulty, namely that the SAVE program does not correctly save .REL files (produced by Microsoft's BAS-COM) loaded by MICROSOFT's L80 relocating loader. I wrote a short BA-

Another in a series of
productivity notes on MS-DOS™
software from UniPress.

**Subject: Multi-window full
screen editor.**

Multiple windows allow several files
(or portions of the same file) to be
edited simultaneously. Program-
mable through macros and the built-
in compiled MLISP™ extension
language.

Features:

- Famed Gosling Version.
- Extensible through the built-in
MLISP programming language and
macros.
- Dozens of source code MLISP
functions; including C, Pascal and
MLISP syntax checking.
- EMACS runs on TI-PC™, IBM-PC AT™,
DEC Rainbow™ or any other MS-DOS
machine. Requires at least 384k.
- Run Lattice® C or PsMake™ in
the background and EMACS will
point to any errors for ease of de-
bugging. PsMake is a UNIX™-style
"make" utility to automate the pro-
cess of building complex programs.
- Optional Carousel Tools: UNIX-
like facilities including cat, cp, cd,
logout, ls, mv, pwd, rm, set, sh
and more.

Price:

EMACS	\$475
One month trial	75
Available for UNIX and VMS.	
PsMake	179
Carousel Tools	149
Full System	1,299
Includes EMACS (object), PsMake, Lattice C, PHACT™ ISAM and Carousel Tools.	

TEXT EDITING

UNIPRESS EMACS™

Subject: Compiler for MS-DOS.

Lattice C Compiler is regarded as
the finest compiler for MS-DOS and
is running on thousands of 8086
systems.

Features:

- Runs on the IBM-PC™ under
MS-DOS 1.0, 2.0 or 3.0
- Produces highly optimized code.
- Small, medium, compact and
large address models available.
- Standard C library.
- PLINK—optional full function
linkage editor including overlay
and support.

Price:

Lattice C Compiler	\$425
PLINK	425

COMPILER FOR THE 8086™ FAMILY

LATTICE® C COMPILER

**Subject: Powerful Keyed File
Access for MS-DOS.**

PHACT ISAM is a keyed B+ tree
file manager providing easy access
to and manipulation of records in
a database.

Features:

- Supports fixed and variable length
records (1-9999 bytes).
- Up to 9 alternate indices are sup-
ported.
- Record locking allows each record
in the database to allow multiple
simultaneous updates.
- Records can be accessed on full
or partial key.
- Includes full Lattice linkable library
and high-level functions.

Price:

PHACT ISAM	\$250
Source Code available, call for terms.	

For more information on these and
other UNIX software products, call or
write: UniPress Software, Inc., 2025
Lincoln Hwy., Edison, NJ 08817.
Telephone: (201) 985-8000. Order
Desk: (800) 222-0550 (Outside NJ).
Telex: 709418. Japanese Distributor:
Softec 0480 (85) 6565. European Dis-
tributor: Modulator SA (031) 59 22 22.

OEM terms available.
Mastercard/Visa accepted.

ISAM FILE SYSTEM

PHACT™

SIC program to list ASCII files, including line numbers; after finishing a listing it prompts the operator for another file name. L80 loads the .REL file, and shows a starting address of 013CH. (The starting address of CP/M programs is 0100H.) If 013C is used as the starting address when the program is saved (using CP/M Plus's SAVE program) it does not run properly. It returns to the operating system after a few seconds, having done nothing visible. (It does not even ask for a file to list.) If the file is saved with a starting address of 0100H, the resulting .COM

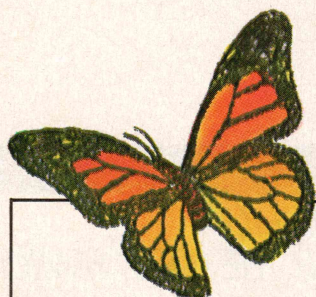
program does execute. It lists the file, and returns to ask for another file when the first file is completed. However, it does not operate properly: with each line it prints a set of characters that do not belong to the file, the same set of characters for each line. If loaded and saved under CP/M 2.2 it operates properly, even when moved to a CP/M Plus diskette. However, if the .REL file is loaded by L80 with the /G option the program operates properly, even under CP/M Plus.

If the .REL file is loaded by Digital Research's LINK, the resulting .COM program executes and lists the file

properly, but hangs without returning to request more input when it has finished listing the file. Recovery requires rebooting the system.

I have written to Digital Research and Microsoft, as well as the vendor from whom I bought the computer, a MAX80, and have not received any assistance. Have you heard about this problem? Or might some of your readers have some information that might be helpful.

Sincerely yours,
Maynard B. Neher
16637 Diaz Drive
San Diego, CA 92128



The Language of Artificial Intelligence

PROLOG V

Includes: interpreter, sample Prolog programs, 122-page manual and primer, and custom-designed binder and slipcase.

ONLY
\$69⁹⁵

Not Copy
Protected

Selected by Japan's Fifth Generation ICOT Project to create machine-based intelligence, Prolog is the most widely used artificial intelligence language worldwide.

POWERFUL AND ELEGANT

With over 70 predefined predicates, PROLOG V complies with the spirit and syntax of Edinburgh Prolog as described in Clocksin and Mellish's classic *Programming in Prolog*, the book that set the de facto standard for Prolog.

IDEAL FOR CREATING:

- ☐ relational data bases
- ☐ expert systems
- ☐ natural language systems
- ☐ biochemical analysis
- ☐ automated design
- ☐ abstract problem solving
- ☐ mathematical logic

THE CHOICE OF UNIVERSITIES

Generous University site licenses have made PROLOG V the choice of Universities from New England to Southern California. Send your inquiry on University stationery, or call for details.

NO RISK OFFER

Examine the package and documentation at our risk for 30 days. Then if you aren't fully satisfied with the quality of the product, return PROLOG V with the distribution diskette still sealed, and we'll refund your money in full.

HARDWARE REQUIREMENTS
☐ PC-DOS/MS-DOS 1.4 or later
☐ 128 K RAM
☐ One disk drive



CHALCEDONY SOFTWARE

5580 LA JOLLA BLVD.
SUITE 126 E
LA JOLLA, CA
92037

☐ PAYMENT ENCLOSED \$ _____
CA residents add 6% sales tax

☐ CHARGE MY: ☐ MasterCard ☐ Visa

Card No. _____ Exp. Date _____

Signature _____

Mr./Mrs./Ms. _____
(please print full name)

Address _____

City/State/Zip _____

PHONE ORDERS:
(619) 483-8513

SHIPPING:
\$ 5.00 U.S.
7.50 Canada
10.00 Caribbean,
Hawaii Air
20.00 Overseas Air

Allow 15 business
days for personal
and company checks.

COD orders not
accepted.

Tiny BASIC

Dear DDJ:

I have enjoyed your journal very much over the years. I especially liked your articles on Tiny BASIC and Small-C. I implemented Tiny-BASIC on the 6800 after your first articles. I wrote a Small-C for the 6809 when you started the Small-C series and did one for the 68000 later. When you published the article on Tiny BASIC for the 68000 in the February issue [DDJ #100 February 1985] I couldn't resist downloading the code and trying it.

I have a suggestion for improvement which I would like to pass along. The RND function does not generate very good random numbers. I wrote the enclosed test program which generates pairs of 1 digit numbers. These should be evenly distributed over the 100 possible combinations. If you try the program you will see that there are several pairs of digits that don't appear at all.

I have enclosed an improved RND function which eliminates this problem. It is only a first try and there is room for improvement in two areas. First it is restricted to a limit of 65535 since I didn't bother to implement a 32-bit multiply. Second it is based on the multiplicative algorithm and I didn't spend any effort picking the best multiplier. It should have a sequence length of 4,294,967,296 however.

John P. Byrns
1953 Governors Ln.
Hoffman Estates, IL 60195
DDJ

Circle no. 21 on reader service card.

The Ultimate Programmer's Editor

WENDIN'S *XTC*™

SUPER PROGRAMMERS edit in XTC to make software development a snap! Just look at these powerful features:



MULTITASKING

XTC's built-in multitasking lets you run your macros in the foreground or independently in the background while you continue editing. A background process has full access to editor resources, and can be used to translate code from one language to another in **REAL TIME**, print files in the background, or even scan syntax while you type in code. Best of all, you can use XTC to edit source and documentation in any programming language!

COMPILE IN WINDOWS

All DOS compilers and utilities can be executed from within XTC using a single keystroke. While it runs, XTC captures your compiler's output and redirects it into your text, so you can compare compiler messages with your source code **ON THE SAME SCREEN**. And using XTC's macro language, Turbo Pascal is literally only a keystroke away. You can use other compilers and utilities inside XTC too — like Lattice "C," Microsoft Pascal, and IBM's Basic, to name a few.

MACRO LANGUAGE

XTC has the most powerful macro language in the editing world. XTC's macros aren't just keystrokes assigned to keys; they're real programs that can be used to automatically edit source code and data files. Like any real programming language, XTC has control structures like **IF THEN ELSE**, **WHILE DO**, **REPEAT UNTIL**, **FOR NEXT**, **DUPLICATE N TIMES**, **INDEFINITE LOOP**, **EXIT**, and **BREAK LOOP**. XTC also has **INTEGER**, **BOOLEAN**, and **STRING** variables to hold numbers, conditions, and pieces of text.

WINDOWS & BUFFERS

With XTC you can display up to 8 different files or parts of the same file on the screen at once. XTC's windows are programmable and can even be linked together to share files. XTC also has 20 other buffers that you can use to hold files and blocks of text.

WORDSTAR COMPATIBILITY

If you already know Wordstar commands, then you don't need to learn a new set of commands. If you want to customize XTC, just write macros to emulate the key layout you're used to. XTC can also read Wordstar files, and can even strip off all of the non-standard high bits with a single command.

LARGE FILE EDITING

XTC lets you edit files entirely in memory (using all available memory), or paged to disk, for maximum flexibility. You can choose how XTC buffers text.

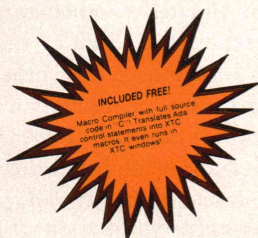
INCREDIBLE EXTRAS!

- UNDO N TIMES
- REMOVE WORDSTAR HIGH BITS
- EDIT GRAPHICS DISPLAYS
- AUTOINDENTING MODE
- TAB EXPANSION/ COMPRESSION MODE
- EXTRA LONG LINES
- MACRO COMPILER
- TELEPHONE SUPPORT

- 150 PAGES OF DOCUMENTATION
- RUNS ON IBM / PC, XT, AND / AT COMPUTERS (AND TRUE COMPATIBLES)

COMPLETE SOURCE CODE

XTC comes with 7,000 lines of source code jam-packed onto two DSDD disks. Includes 13 modules written in Pascal, and 2 assembly libraries you can use to access the PC's screen, intercept software interrupts (like INT 21H functions), allocate and deallocate memory, and load and execute programs. It's all included **FREE** for your recreation and enjoyment!



ORDER HOTLINE
509/235-8088
CREDIT CARDS WELCOME!

XTC outperforms any other programmable editor on all IBM/PC, /XT, and /AT computers (and true compatibles). If you want to feel the power of XTC before you buy it, just ask for our demo disk (only \$10) and try it out. When you buy XTC, we'll knock 10 bucks off the price.

To get your copy of XTC now, order it over the phone — we can ship it the same day you call! Or, send in an order, just like this one:

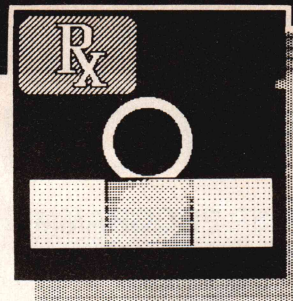
XTC \$99.00
Macro Compiler FREE
Shipping, Handling,
Insurance 3.50
Want it COD? Add this 1.90

TOTAL IT UP, AND SEND IT QUICK!

WENDIN™
BOX 266
CHENEY, WA 99004

The people who make quality software tools affordable.

Ada² is a registered trademark of the U.S. Department of Defense. Turbo Pascal is a trademark of Borland, Inc. XTC is a trademark of Wendin, Inc.



by D. E. Cortesi

Random Moves

We are still collating the letters we've received on that pseudo-random number generator (PRNG) we showed in February. Don't use it. We promise emended and improved versions in Forth and assembly language next month.

Turbo and CP/M Plus

The estimable Turbo Pascal product has a design error that makes its 8-bit version unusable under CP/M Plus (CP/M 3.0) and dubious under CP/M 2.2. When you compile a program to form a .com file, Turbo includes several kilobytes of runtime support in the output, along with the code generated from your Pascal source. That's not the problem; runtime support code is necessary.

Part of the job of that runtime library is to initialize the program's environment when it is loaded and begins execution. Anyone who has written assembly code for CP/M knows what must be done: the stack pointer must be set up and the size of available storage discovered. The usual way of sizing storage is to load the BDOS entry address from location 6.

It's a universal CP/M convention that system programs can reserve storage for themselves at the top of the program area by reducing that address. DDT and SID do it; XSUB does it; spoolers and key-macro programs do it; and CP/M Plus has elevated it from an informal convention to an elaborate system of Resident System Extensions (RSXs). As a result, the top of storage is a dynamic value in CP/M. It can change from day to day if the BIOS expands and from moment to moment as system programs are loaded.

The Turbo compiler, however, treats the size of storage as a con-

stant. At compile time, it places in the runtime code a constant number reflecting the size of the program as the compiler found it; when the .com is executed, it expects storage to be exactly the same size. If programs compiled under Turbo V.3 find less storage than the compiler had, they abort with the message "Not enough memory." This has nothing to do with how many bytes the program needs. Indeed, our test program didn't need any; it simply wrote "hello" 10 times. If we compile it when no RSXs are present but run it when one is, it aborts. Run under SID, it aborts. It's a strange oversight in an otherwise solid product.

Turbo and Standard Pascal

Understand, please, that we *like* Borland and its marketing policies. Its products give good value for the money, and Borland backs them with support that is much better than average. Between Borland and the "shareware" pioneers, a whole new level of price/performance is being established in the software industry. So spare us the letters about how Borland has done so much for software, etc., etc. We appreciate all that.

Turbo Not Standard

But we are getting just a tad tired of the claim that Borland's Turbo Pascal is "standard." It is definitely *not* standard Pascal, which, in a compiler that seems destined to dominate the micro world, is a damn shame. Its price and good user interface commend it for school use, but we certainly couldn't recommend it for that purpose, and we doubt any college computer science department would accept it despite its friendliness.

Let's talk about the Pascal stan-

dard, how Turbo violates it, and what that costs its users.

The Ignored Standard

There *is* a Pascal standard. In the U.S., it is sanctioned by the IEEE Computer Society and the American National Standards Institute (ANSI); abroad, by the International Standards Organization (ISO). Although ANSI finalized it in December 1982, it was available at least two years earlier in drafts that differed from each other in only the minutest points and that nobody expected would change in any significant way prior to final approval.

The standard language omits features that many people want. Although it defines a minimum Pascal, it explicitly opens the door to extensions. It defines an extension as "a modification to . . . this standard that does not invalidate any program complying with this standard . . . except by prohibiting the use of one or more particular spellings of identifiers." In other words, you may add any features you like to Pascal as long as you compile standard programs correctly. The last clause even lets you add new reserved words to the language, words that could be user-defined names under the standard.

The standard lays two burdens on the developer. First, you must document variations from the standard in certain ways; that's usually no problem. The other is that a translator must "be able to determine whether or not a program violates . . . this standard . . . and report the result of this determination to the user." In other words, you must warn the user when a program is not standard Pascal. That's essential to portability; otherwise what you intend to be a standard (hence, portable) program can slip

into an unstandard state that is undetectable until you attempt to port it.

Standard Pascal is a small language but still bigger than, say, Fortran IV; more flexible than, say, COBOL; and easier to teach, to use, and to implement than, say, Ada. If there were general compliance with the standard, we could teach people a single language they could use anywhere, at least for small and medium problems. If you moved to a new implementation, you would have to learn the peculiar extensions it supported, but you could still do useful programming while you learned. And programs you wrote using the standard language would be portable anywhere.

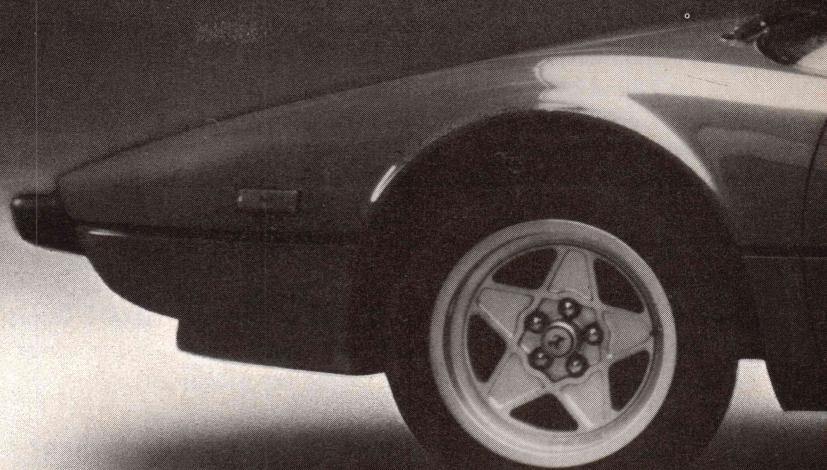
It's ironic that many newer and less forgiving standards (e.g., for graphics, for terminal escape sequences, for various buses) garner wide respect and conformance, while the modest and unconfining Pascal standard, draws only scorn from developers.

Turbo Versus the Standard

The Turbo manual (*Turbo Pascal Version 3.0 Reference Manual*, Borland International, Inc., 1985) doesn't contain the kinds of documentation called for by the standard, but it does have Appendix D, "Turbo vs. Standard Pascal" (p. 319). This shows at once that Borland is unaware of the ANSI/ISO standard. "The Turbo Pascal language," it says, "follows the Standard Pascal defined by Jensen & Wirth in their *User Manual and Report* . . ."

We hate to disturb such blissful slumber, but the language has changed in the 10 years since the *User Manual and Report* was published. The ANSI/IEEE committee convened in 1979 and published its first draft standard in 1980, two years before Turbo was born.

Turbo follows its outdated standard "with only minor differences introduced for the sheer purpose of efficiency." Efficiency for whom or what? For users who want to port programs or to port their hard-won knowledge of the language? Surely not! What then? Efficiency in the execution of the generated object programs? We will examine the case for that as we go; it doesn't hold up.



Finally, A Lint and Make for MS™ - DOS

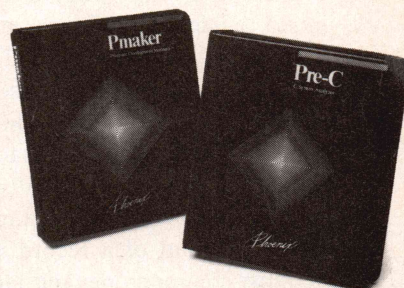
Get the full range of features C programmers working in UNIX™ have come to expect from their Lint and Make utilities. With Pre-C™ you can detect structural errors in C programs five times faster than you can with a debugger. Find usage errors almost impossible to detect with a compiler. Cross-check multiple source files and parameters passed to functions. Uncover interface bugs that are difficult to isolate. All in a single pass. Capabilities no C compiler, with or without program analyzing utilities, can offer. Pre-C outlits Lint, since you can handle analyses incrementally.

Pre-C's flexible library approach lets you maintain continuity across all programs in your shop, whether you use Pre-C's pre-built libraries, functions you already have, or some you might want to buy.

Plus, you're not limited to one particular library. Pre-C keeps track of all the libraries you're using to make sure that code correctly calls them.

With Pmaker™ you can update and track every module in your program. When you make a change in any source or include file, all you do

is run Pmaker. It will recompile changed modules and relink your program. With any compiler or linker you choose. Pmaker can update an object module library when one or several of the object modules are changed. You can use Pmaker to handle any task when a change requires several steps.



Pre-C by Phoenix. \$395. Pmaker by Phoenix. \$195.

Call (1) 800-344-7200.
In Massachusetts (617) 762-5030.

Or, write: Phoenix Computer Products, Corp., 1420 Providence Highway, Suite 115, Norwood, MA 02062.

Phoenix

PROGRAMMERS' PFANTASIES™ BY PHOENIX

Programmers' Pfantasies, Pre-C, and Pmaker are trademarks of Phoenix Computer Products Corporation. MS is a trademark of Microsoft Corporation. UNIX is a trademark of AT&T Bell Laboratories.

Could the changes be to improve the efficiency of the compiler itself? Very possibly; we'll see that the elimination of some hard cases may have let the compiler be a little simpler.

Or could the manual be referring to the efficiency with which the developers can bring a product to market? Did Turbo drop the hard parts so the developers could finish their compiler more quickly?

New and Dispose

Turbo v.1.0 skipped the standard way of managing dynamic storage, the procedures *New* and *Dispose*. V.2 added them but in incomplete form. Turbo supports the basic features: *New(ptr)* creates a new one of whatever *ptr* is declared to address, and *Dispose(ptr)* reclaims the space used by whatever *ptr* points to.

The standard allows a special case for allocating variant records (records that may have different lengths depending on a tag field). Suppose that *ptr* bases a variant record type and that *tag* is one of the constants that select among the variants. *New(ptr)* will allocate space to hold the longest variant, but *New(ptr,tag)* should allocate just enough space to hold the *tag* variant. *Dispose(ptr,tag)* is then required to release such a record.

Turbo omits these special forms. Why? Not to enhance the user's efficiency; it will seriously complicate your life if you try to import to Turbo a program that uses this feature. Nor for the program's efficiency; the shortest variant of a record is often the most common one, and lacking a way to allocate short records, you can waste a lot of dynamic storage—sometimes enough to make a program infeasible.

Turbo didn't have to ban these cases. It could have accepted them while ignoring the *tag* parameters. Then standard programs would at least compile. But why not support them in full? The standard says *tag* must be a constant; therefore, Turbo could still tell at compile time how many bytes *New* allocates or *Dispose* releases. The extra compiler code couldn't be large, and the cases have no effect on the runtime library.

But the manual adds insult to inju-

ry. "The restriction," it chirps, "is easily circumvented by using the standard procedure *GetMem*." Wrong on every count! *GetMem* is not "standard"; that name is unique to Turbo. Nor is *GetMem* type-secure, as *New* is. Plus it requires a count of bytes to allocate, so to use it you must know in detail how Turbo allocates space in variant records. Then heaven help you if you change the record type but forget to change the *GetMem* calls! (Turbo has a non-standard *SizeOf* function, but it won't give you the size of a particular record variant.)

The Page Procedure

The standard specifies a list of procedures and functions that a compiler must predefine. Turbo has all the trigonometric functions, all but two of the I/O procedures (discussed below), and dozens of nonstandard routines for "turtle" graphics, special I/O operations, DOS calls, and so on.

But it lacks the simple procedure *Page*. *Page(f)*, where *f* is a text (ASCII) file, is supposed to append a newline if the file isn't currently at the head of a line then "cause an implementation-defined effect on the textfile *f*, such that subsequent text . . . will be on a new page if the textfile is printed on a suitable device." Like the other Pascal I/O routines, *Page* with no parameter works on the file output by default.

Page may not seem like much of a loss, but its absence puts the user in an infuriating bind. Like the other Pascal I/O procedures, *Page* cannot be duplicated by user code. User procedures can't have default parameters, and there is no way to perform the essential test of a file variable to see if it is at the head of a line. (A *Page* that always writes a newline will sometimes force an extra blank sheet where the standard *Page* will not.)

What do you do to import a standard program that relies on *Page*? You find every use of *Page* and change it to call one of two procedures that you must write yourself—procedures that can't duplicate the standard actions perfectly. If the output isn't satisfactory, you must dig into the program's logic. Changing

the logic of an imported program, one written perhaps by total strangers, because of a stupid omission from a compiler is not a good use of time. If you write a program that you will later export, you must include your homebrew *PageNamedFile* and *PageDefault* procedures, surrounded with big comment blocks explaining how to alter them to use *Page* if it should be available.

What would *Page* cost: 64 bytes of code in the runtime library and maybe 32 bytes of table space in the compiler? Its overhead would be undetectable in a compile and zero at runtime. Why did Turbo omit it? We'll never know. Appendix D says it's because "the CP/M operating system does not define a form-feed character," but that is pure waffle. If there is one thing common to all CP/M systems, it's ASCII, whose form feed is respected by every printer we know.

Get and Put

Wirth gave a mathematician's definition of I/O. He began with a rigorous definition of the file data type. Then he chose a minimal set of operators on that type, operators whose actions are regular enough to be useful in formal verifications of program correctness. Finally he defined all file operations in terms of these minimal operators. The standard follows the *User Manual and Report* in giving a rigorous logician's definition of file I/O, all based on the operators *Get* and *Put*.

What do these fundamental operations do? First, understand that to Wirth a file variable is a species of pointer variable. He says that a file is a sequence of objects, while a file variable is a pointer to only one of the objects in the sequence. The declaration

fr : file of real;

asserts two things: somewhere there is a sequence of real numbers (possibly empty), and under the right circumstances *fr* will address one of them. The statement

Reset(fr);

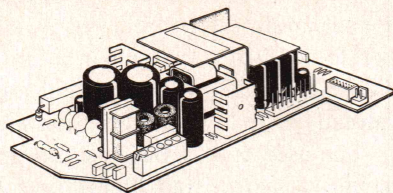
sets up the circumstances; *fr* now ad-

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

Switching Power Supply!

- + 5VDC - 8 Amps
- +12VDC - 5 Amps
- 12VDC - 1 Amp
- (81 WATTS MAX)



\$29⁹⁵
ea.

4 FOR \$99

ADD \$2
EA. UPS

BRAND NEW UNITS, MFG. BY BOSCHERT FOR HEWLETT PACKARD! PERFECT FOR SMALL COMPUTER AND DISK DRIVE APPLICATIONS #XL81-5630. INPUT 110V/220V, 50/60 HZ. NOMINAL OUTPUTS: +5VDC @ 8A, +12VDC @ 5A, -12VDC @ 1A. TOTAL MAX OUTPUT. MUST BE LIMITED TO 81 WATTS TOTAL! (WITH PIN OUT SHEET.) ORIGINAL FACTORY BOXED.

64K \$100 STATIC RAM

\$139⁰⁰
KIT

NEW!

LOW POWER!

150 NS ADD \$10

BLANK PC BOARD
WITH DOCUMENTATION
\$49.95

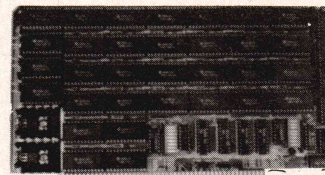
SUPPORT ICs + CAPS
\$17.50

FULL SOCKET SET
\$14.50

FULLY SUPPORTS THE
NEW IEEE 696 \$100
STANDARD
(AS PROPOSED)

FOR 56K KIT \$125

ASSEMBLED AND
TESTED ADD \$50



FEATURES: PRICE CUT!

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports IEEE 696 24 BIT Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- * 2716 EPROMs may be installed in any of top 48K.
- * Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- * Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- * BOARD may be partially populated as 56K.

64K SS-50 STATIC RAM

\$119⁰⁰
(48K KIT)

NEW!

LOW POWER!

RAM OR EPROM!

BLANK PC BOARD
WITH
DOCUMENTATION
\$52

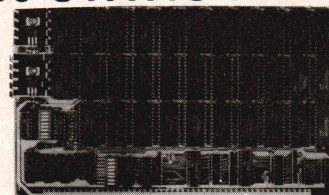
SUPPORT ICs + CAPS
\$18.00

FULL SOCKET SET
\$15.00

56K Kit \$129

64K Kit \$139

ASSEMBLED AND
TESTED ADD \$50

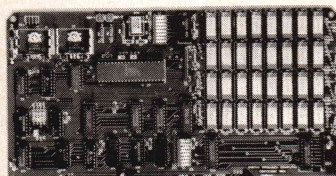


FEATURES:

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- * 2716 EPROMs may be installed anywhere on Board.
- * Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- * One Board supports both RAM and EPROM.
- * RAM supports 2MHZ operation at no extra charge!
- * Board may be partially populated in 16K increments.

256K S-100 SOLID STATE DISK SIMULATOR!
WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



BLANK PCB
(WITH CP/M* 2.2
PATCHES AND INSTALL
PROGRAM ON DISKETTE)
\$69⁹⁵
(8203-1 INTEL \$29.95)

FEATURES:

- * 256K on board, using + 5V 64K DRAMS.
- * Uses new Intel 8203-1 LSI Memory Controller.
- * Requires only 4 Dip Switch Selectable I/O Ports.
- * Runs on 8080 or Z80 S100 machines.
- * Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- * Provisions for Battery back-up.
- * Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
- * The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- * Compare our price! You could pay up to 3 times as much for similar boards.

(ADD \$50
FOR A&T)

\$169⁰⁰

#LS-100

(FULL 256K KIT)

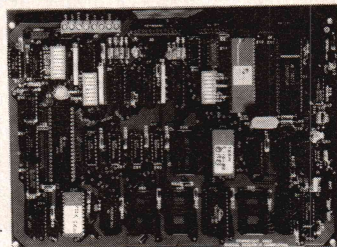
THE NEW ZRT-80

CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:

- * Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- * RS232 at 16 BAUD Rates from 75 to 19,200.
- * 24 x 80 standard format (60 Hz).
- * Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- * Higher density formats require up to 3 additional 2K x 8 6116 RAMs.
- * Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- * 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- * Composite or Split Video.
- * Any polarity of video or sync.
- * Inverse Video Capability.
- * Small Size: 6.5 x 9 inches.
- * Upper & lower case with descenders.
- * 7 x 9 Character Matrix.
- * Requires Par. ASCII keyboard.



\$89⁹⁵

#ZRT-80
(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716
CHAR. ROM. 2732 MON. ROM

\$49⁹⁵

SOURCE DISKETTE - ADD \$10

SET OF 2 CRYSTALS - ADD \$7.50

FOR 8 IN. SOURCE DISK
(CP/M COMPATIBLE)
ADD \$10

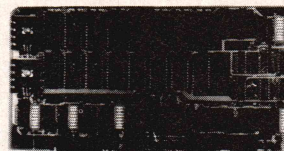
32K \$100 EPROM/STATIC RAM

NEW!

FOUR FUNCTION BOARD!

NEW!

EPROM II
FULL
EPROM KIT
\$69.95
A&T EPROM
ADD \$35.00



BLANK
PC BOARD
WITH DATA
\$39.95

SUPPORT
ICs
PLUS CAPS
\$16

FULL
SOCKET SET
\$15

We took our very popular 32K \$100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

FEATURES:

- * This one board can be used in any one of four ways:
A. As a 32K 2716 EPROM Board
B. As a 32K 2732 EPROM Board (Using Every Other Socket)
C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
D. As a 32K Static RAM Board
- * Uses New 2K x 8 (TMM2016 or HM6116) RAM's
- * Fully Supports IEEE 696 Buss Standard (As Proposed)
- * Supports 24 Bit Extended Addressing
- * 200 NS (FAST!) RAM's are standard on the RAM Kit
- * Supports both Cromemco and North Star Bank Select
- * Supports Phantom
- * On Board wait State Generator
- * Every 2K Block may be disabled
- * Addressed as two separate 16K Blocks on any 64K Boundary
- * Perfect for MP/M* Systems
- * RAM Kit is very low power (300 MA typical)

PRICES
SLASHED!

32K STATIC RAM KIT — \$109.95

For RAM Kit A&T — Add \$40

Digital Research Computers

P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add \$3.00 postage. We pay balance. Orders under \$15 add 75¢ handling. No. C.O.D. We accept Visa and MasterCard. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50. add 85¢ for insurance.

dresses the first real number in the file. For comparison, recall that

```
pr : ^real;
```

sets up a variable *pr* that points to a single real number, as in

```
x := 3.5 * pr^;
```

Therefore, *Reset(fr)* sets things up so that *fr* addresses a real number:

```
x := 3.5 * fr^;
```

Thus did Wirth unify the treatments of I/O and dynamic storage.

Standard Get and Put

The *Get* procedure advances a file one unit in its sequence. After *Get(fr)*, *fr* addresses the next real number in the sequence.

Rewrite opens a file for output; the statements

```
nr : file of real;  
Rewrite(nr);
```

create an empty sequence of real numbers and make *nr* point to a real of undefined value. You may use *nr* exactly like any other pointer to reals:

```
nr^ := sqr(pr^) + sqr(fr^);  
pr^ := sqrt(nr^);
```

The statement

```
Put(nr);
```

appends the current referent of *nr* to its file and makes *nr* point to undefined space again.

Do you find these ideas confusing, difficult, baffling? Do you think they make for an inflexible system? Or do you agree that they are simple, logical, consistent, and highly useful?

The Borland programmers did not. Turbo does not support *Get* and *Put*, nor does it permit a file variable to be used as a pointer. It offers only the *Read* and *Write* procedures (which to Wirth are composite operations defined in terms of *Get* and *Put*).

Simplicity?

Appendix D offers four excuses for

the omission. One is that *Read* and *Write* are “easier to understand.” A computer science teacher would never say that. Good teachers explain the basis of things. You cannot explain the basis of *Read* and *Write* without referring to *Get* and *Put*; you can state their actions as arbitrary rules to learn by rote, but you can’t explain them.

Of course, if you are already an experienced programmer—one used to Fortran or BASIC, perhaps—you might find the *Read* and *Write* procedures more *familiar* than *Get* and *Put*, and you might confuse familiarity with simplicity. That’s a common error, as anyone will attest who has tried to explain the arbitrary, but totally familiar, rules of English to a foreigner.

Versatility?

One of the three other excuses is that *Read* and *Write* are “far more versatile” than *Get* and *Put*. That’s rubbish, but our hypothetical ex-Fortran user wouldn’t realize it without close study of Pascal.

With *Put*, you can assign and reassign an output value as often as you like; the value isn’t committed to the file until *Put* is issued. That concept isn’t in the thinking of a PL/I user; in most languages, once you associate data with a file, it’s gone. Once you comprehend the idea, however, you can find lots of uses for *writing* data without *putting* it.

Get is even more useful. If you’ve read Kernighan and Plauger’s *Software Tools*, you may recall how often they resort to the *ungetc()* procedure to replace the character last read from a file. With Pascal standard input, you can look at the next value coming from a file without committing yourself to removing it from the file. You simply test the value addressed by the file variable. If you want it, you assign it somewhere else and issue *Get*. You may also leave it for another part of the program to use, or discard it with *Get*. Here’s a procedure, free of side effects, to strip blanks from an input file:

```
procedure fstrip(var f: text);  
begin  
  while (not eof(f))
```

and (*' '=f^*) do

```
  Get(f)
```

```
end;
```

Afterward, either the next element of the file is a nonblank or the file is finished. Try writing a comparable function in C or BASIC—or in Turbo! It takes either *ungetc()* or your own input function with a static variable.

Because *Get* and *Put* are standard and highly useful, you may assume that Pascal programmers make use of them. A lot of Pascal programs are floating around on minis and mainframes and in textbooks. The odds are good that if you try to import one to Turbo, you’ll fail because the program uses *Get* or *Put*. You may expect to find as well that the assumptions behind *Get* and *Put* are woven deeply into the program’s logic; most programs will need major changes to run without them.

Less Overhead?

Appendix D offers the excuse that, without *Get* and *Put*, “variable space overhead is reduced, as file buffer variables are not required.” True, the compiler would have to allocate a buffer the size of one file data item in addition to the sector buffer it needs for DOS I/O. Alternatively, it could round the size of a data item up to a whole number of DOS sectors, add one sector, and allocate a single buffer of that size (this ensures a complete data item will always fit in the buffer).

Banning standard use of the file variable lets the compiler get by with a single sector buffer. Does that reduce “variable space overhead”? Of course not! Now *you* have to define the variables! You need extra variables to hold data items before they are written to the file, instead of merely assigning and reassigning until a *Put* is done. You need variables to hold items after reading them from the file and before using them in expressions. And if it turns out the items shouldn’t have been read, Turbo offers no *ungetc()*, so *you* get to define one—again adding code overhead.

Speed?

Appendix D’s premier excuse is that “*Read* and *Write* give much faster

I/O." Do they? True, a naive implementation of Get might often copy data from a sector buffer to an item buffer. But Read *requires* copying data to a user-defined variable, whereas a smart implementation of Get could avoid moving data at all. Many times when the file data type is smaller than half a sector (the typical case), Get need do no more than increment a pointer, check for end of sector, and return. That's quick!

If a new sector must be read, a Get implementation might have to move part of the last sector from the end of the buffer to its head before reading more data. But that move always involves *less* than a single data item! Under MSDOS, when the file data type is composite (record, set, or array), a clever compiler would push the whole burden onto the DOS. It would allocate no sector buffer at all, just an item buffer, and use "handle" I/O to read or write only as many bytes as needed.

A good implementation of Get and Put might entail 15 percent more code than the present Turbo library, but by moving less data, it might well execute in 25 percent less time.

Procedure Parameters

One of the least-regarded features of standard Pascal is the ability to pass a procedure or function as a parameter. Let's say "routine" instead of "procedure or function." Then the feature sounds simpler: the ability to pass one routine the name of another. Its purpose is to let a routine's behavior be parameterized along with its data. Unfortunately, neither Jensen nor Wirth nor the standards committee could find a good example of its use except "finding the minimum of a function by bisection." Now really, what can an honest programmer say to that but "Gah, Wha?"—meaning, surely you can omit such an abstruse feature from a language without anyone's noticing!

Let's try to suggest some examples to show why it should stay. The first might be a generalized sort that takes, as a parameter, the name of a function that compares two data items and returns True if one is less than or equal to the other. Now de-

fine two comparison routines:

```
function Up(a,b:real):Boolean;
begin Up := (a <= b) end;
```

```
function Down(a,b:real):Boolean;
begin Down := ( b <= a ) end;
```

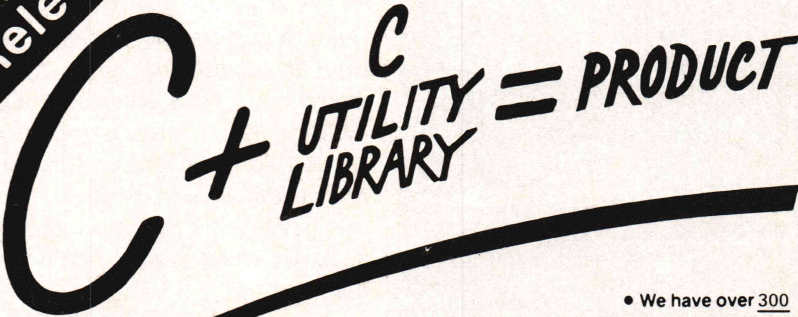
The identical generalized routine will sort ascending if we pass it the name of Up and descending if we pass it Down.

Notice that if we make Up and Down take, not the data to be com-

pared, but ordinal indices to the data, and if we also pass a procedure Swap that interchanges data items given their indices, we will have removed all dependencies on data type from the general sort routine. We need only pass it the Up or Down and the Swap that are appropriate to the data type we want to sort.

As another example, consider a situation where a service routine can detect an error, for which the correct recovery action depends on the conditions under which the routine was

New Release



- We have over 300 complete, tested, and, documented functions. All source code and demo programs are included.
- The library was specifically designed for software development on the IBM PC, XT, AT and compatibles. There are no royalties.
- Over 95% of the source code is written in C. Experienced programmers can easily "customize" functions. Novices can learn from the thorough comments.

We already have the functions you are about to write

Concentrate on software development—not writing functions.

THE C UTILITY LIBRARY includes:


- Best Screen Handling Available • Windows • Full Set of Color Graphics Functions • Better String Handling Than Basic • DOS Directory and File Management • Execute Programs, DOS Commands and Batch Files • Complete Keyboard Control • Extensive Time Date Processing • Polled ASYNC Communications • General DOS BIOS gate • Data Entry • And More •

- The Library is compatible with: Lattice, Microsoft, Computer Innovations, Mark Williams and DeSmet. Available Soon: Digital Research, Aztec and Wizard.

C Compilers: Lattice C—\$349, Computer Innovations C86—\$329; Mark Williams C—\$449.

C UTILITY LIBRARY \$185. Special prices on library & compiler packages.

Order direct or through your dealer. Specify compiler when ordering. Add \$4.00 shipping for UPS ground, \$7.00 for UPS 2-day service. NJ residents add 6% sales tax. Master Card, Visa, check or P.O.



ESSENTIAL SOFTWARE, INC

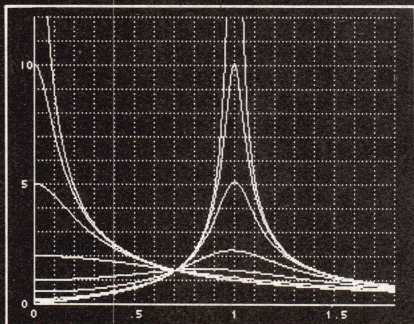
P.O. Box 1003 Maplewood, New Jersey 07040 914 762-6605

Circle no. 36 on reader service card.

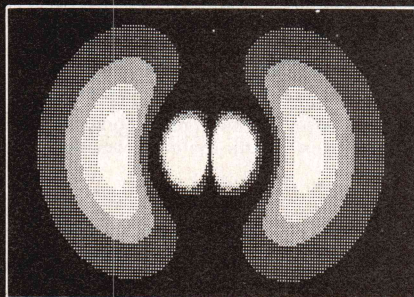
isys FORTH

for the Apple®][

Fixed point speed can rival that of floating point hardware. But the details have been a well kept secret—until now. The following graphs were generated by fixed point examples from the ISYS FORTH manual.



Parallel Resonance with Damping
BASIC 213 sec ISYS FORTH 27 sec



Hydrogen 3p Orbital Cross-section
BASIC 492 sec ISYS FORTH 39 sec

- **Fast native code compilation.** Sieve benchmark: 33 sec
- **Floating Point**—single precision with transcendental
- **Graphics**—turtle & cartesian with 70-column character set
- **Double Precision** including D*/
- **DOS 3.3 Files** read & written
- **FORTH-83** with standard blocks
- **Full-Screen Editor**
- **Formatter** for word processing
- **Macro Assembler**
- **Price: \$99**, no extra charges

ILLYES SYSTEMS

PO Box 2516, Sta A
Champaign, IL 61820

Technical Information:
217/359-6039, mornings

For any Apple][model, 48K or larger.
Apple is a registered trademark of Apple Computer.

called. There are three ways to handle this: (1) You can write a version of the routine for each recovery method; (2) you can pass the routine an error-recovery action number and put a big error-handling case statement in it; or (3) you can pass the routine the name of an error-handling procedure. Each client may then contain a local procedure to do error recovery in its own style. The service routine remains general, while the recovery actions are localized in those parts of the program that establish the need for them.

Turbo Pascal is not alone in banning procedure parameters; it's the most ignored feature of the language. Why? It seems like a fairly simple thing to implement—at first. At runtime, it amounts to just pushing the address of a routine on the stack. At compile time, the compiler must check that the parameter list of the routine that is passed matches the parameter list that the receiving routine declared, which is easier to do than to describe.

The real problem is the implementation of Pascal's rules for the scope of variable names. When procedure parameters are forbidden, the dynamic scope of a name matches its lexical scope; the compiler can tell from the program text exactly what names are accessible at any point in the program. When procedure parameters are allowed, cases occur where the referent of a name can be resolved only by extra stack linkages. That complicates the runtime code for stack management and for procedure call and exit.

Although Borland may seem to have made a good tradeoff in dropping procedure parameters, the usual arguments can be made for the feature: it's useful and likely to appear in published programs, especially in the more sophisticated ones, those that are at once most valuable and hardest to modify. We might add that, inasmuch as Turbo threatens to be the single most common implementation of Pascal ever, Borland has an obligation to support the whole language.

The stack management is not that complicated anyway. Full Pascal stack linkage no doubt would slow

down an 8080 or Z80 implementation (for once there's a legitimate reason for restricting the CP/M version!), but that is not true of the 8086.

What would be the penalty for full Pascal in the MSDOS version of Turbo? The compiler would be slightly more complicated, as noted. And the runtime code? Studies have shown that the overwhelming majority of references are to local and global names and to parameters. The troublesome references, to names defined in intermediate scopes, are very rare. (It's tempting to modify the scope rules to forbid just these references, but that would create incredibly subtle portability bugs.) If the generated code optimizes common usages and penalizes the rare ones, the cost should be small.

Summary

The Pascal standard defines a nice little language, but despite the language's popularity and the standard's age, the features of the language and the liberality of the standard are still misunderstood or unappreciated. The justifications in the Turbo manual's Appendix D just don't hold water; likely the real reason Turbo is non-standard is that the Borland programmers were in a hurry to get to market so they dropped some features that were obscure (to them) or of little use (to people who don't understand Pascal well). Yet Turbo Pascal has many admirable features. If it added to them full compliance with the Pascal standard, it would be without question the finest implementation of Pascal for personal computers and among the finest on any computer.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 190.

We're getting hardnosed at Softway.

From now on MATIS™ is only \$49⁹⁵!

(MATIS, the complete User Interface development tool has been selling for \$150.)

Why the radical price cut?

We decided after looking over the competition that MATIS had so many advantages it should be made available to more programmers. We decided to compete aggressively so you could easily afford to have MATIS in **your** bag of tricks. We hear from MATIS users in the USA and France that it is a truly loveable product. Sooo...we're running this big ad to promote our new low price.

MATIS windows are beautiful.

Display any portion of any screens you create at any point in your program. Scroll in any direction manually with cursor keys...or automatically.

And the screens are HUMUNGEOUS!

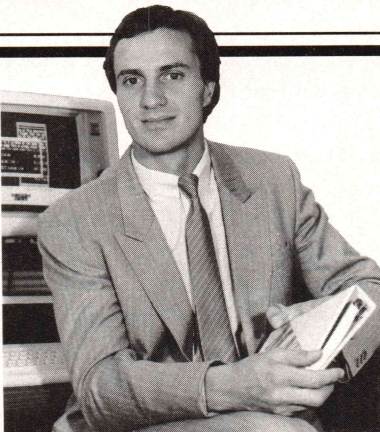
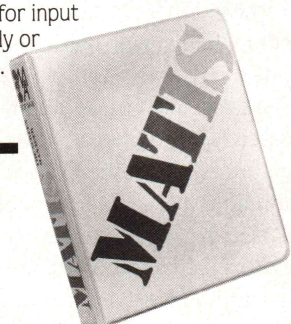
MATIS screens can be just about as big as you want...up to 65,534 rows by 65,534 columns! The number of screens is only limited by available memory.

Print big MATIS screens directly.

One command sends your screens to your printer with no need to program special routines when your virtual screen is bigger than your terminal screen.

User input fields are a snap.

Creating fields for data entry is easy with no limit to size or number by screen. Request for input separately or in blocks.



Denis Moran
President, Softway, Inc.

Control your keyboard with MATIS.

It keeps track of keys that are pressed during the execution of your program and lets you assign specific functions to selected keys.

Control that screen too!

MATIS is extremely versatile and flexible when it comes to controlling lines, columns, fields, and text. They can be modified, transferred, displayed or moved with a single command. All video attributes are supported: color, reverse video, blinking...you name it, you got it.

Want an interactive screen builder?

You've got it with MATIS. It's called "MATPAGE"™ and it lets you create and modify any of your screens in an interactive mode.

MATIS adds over 70 routines to your program.

Written in Assembler, MATIS routines are fast and powerful giving your program improved efficiency and enhanced visual appeal, while they reduce its size and maintenance worries. And MATIS separates screen design from the core of your program.

MATIS is unique.

We don't think there's a single program that combines as many tools in one package as completely or as well as MATIS. It interfaces with Interpreted and Compiled BASIC (Microsoft), C (Lattice, Microsoft, Aztec), PASCAL (IBM, Microsoft) and ASSEMBLER. All you need is an IBM* PC/XT or true compatible under DOS, 128k or RAM, monochrome or color monitor.

You get an easy to follow no-frills manual and a 30-Day Money Back Guarantee.

Late News:

MATIS/T™ for TURBO-PASCAL only \$29.95**

An indispensable add-on at a dynamite price. What more can we say?

Denis Moran
Denis Moran

*MATIS, MATIS/T, & MATPAGE are Trademarks of Softway, Inc.

*IBM is a Reg. Trademark of IBM Corp.

**Turbo Pascal is a Reg. Trademark of Borland International

Softway, Inc.

24-Hour Credit Card Orders By Phone:

1 (800) 227-2400 EXT 989 In California: 1(800) 772-2666 EXT 989

Please ship the following at once. I understand there is a 30-day money back guarantee.

_____ Copies of MATIS at \$49.95 plus \$3 shpg. \$ _____

_____ Copies of MATIS/T for TURBO PASCAL at \$29.95 plus \$2 shpg. \$ _____

California residents, add 6½% sales tax \$ _____

☐ I like to read specs, so send me a folder. Total \$ _____

Name _____

Address (please no P.O. Box) _____

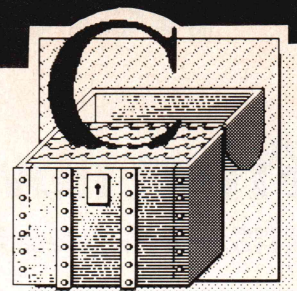
City/State/ZIP _____

Phone (_____) _____ Signature _____

Make payment by money order, check, or charge card ☐ VISA ☐ MASTER CARD

Number _____ Exp. _____

Distributed in Europe By: MICRO APPLICATION SOFTWARE • 147 Avenue Paul Doumer, 92500 RUEIL MALMAISON FRANCE, Tel (1) 732.92.54



by Allen Holub

As part of my ongoing struggle to turn MSDOS into Unix, I write a lot of Unix-like utilities. This month we'll look at a better MSDOS directory utility called "ls" (after the Unix model). Ls uses several subroutines useful in their own right. A multicolumn print utility and an MSDOS interfacing utility (which corrects various deficiencies in the one supplied by Lattice) are two of these.

How To Use Ls

Ls prints a sorted directory in a multicolumn format. The columns are read like a magazine; that is, an entire column is read from top to bottom, then the next column is read. Ls prints volume labels, which are in boldface, first. Next it prints directories; they are underlined. Then it prints the files. Finally, it prints the amount of space used by the directory (instead of the space left on the disk), along with a file and directory count. The figure (page 21) shows various sample outputs from ls.

Usage is: `ls [options] <file list>`. The legal command line options are:

-a

List all files, including hidden files. Like Unix, all files whose names begin with a period (.) are considered invisible (even if the hidden attribute bit isn't set); -a causes these files to be printed too.

-c<num>

Print the output in <num> columns instead of the default five. Up to six columns can fit on a screen. The command

```
ls -cl *.c >foo.bat
```

is useful for making batch files. No spaces are allowed between -c and the number.

-d

List only directories (no files).

-e

Sort by extension and then by filename. This groups all the .c files together and all the .obj files, and so on. The filenames are sorted within their extension categories (i.e., bar.c precedes foo.c, and both precede bar.obj).

-f

List only files (no directories).

-l

List in long format. -l lists the file size, the create time and date, and the file's attributes. Possible attributes are:

D - (D)irectory

H - (H)idden

L - Volume (L)abel

M - File has been (M)odified since last backup

R - (R)ead only

S - (S)ystem

-s

Suppress directory name-underlining and label-boldfacing so that you can direct the output to a printer without the extra escape sequences. Note that directories sort to the top of the list because the graphics require an ESC (which has a smaller value than any letter) at the beginning of the name string. Because the ESC is no longer present, the directory names will be mixed in with the filenames when you use -s.

-u

Print directory unsorted.

You may combine options (`ls -ac3 *.c`) or list them separately (`ls -a -c3 -s *.c`). You can place them anywhere on the command line (`ls *.c -l` is the same as `ls -l *.c`).

The <file list> is a list of the files you're looking for. You can list files explicitly (`ls foo.c bar.c`) or with ambiguous references (`ls *.c`). A directory may precede the filename (`ls`

`\src\tools\ls.c`). You can use / as well as \ to separate directory names (`ls /src/tools/ls.c`). If you request a single directory, then ls lists the contents of that directory. If no <file list> is given, ls prints all files in the current directory.

To make the underlining and boldfacing work, you must find the line "device=ANSI.SYS" in your config.sys file when the system boots. If you don't do this, your screen will look funny unless you use the -s option.

Getting a Directory

You can access MSDOS directories in two ways. A set of low-level DOS function calls "Search for First Entry" (0x11) and "Search for Next Entry" (0x12) work much like the equivalent CP/M functions. (An example of CP/M directory accessing appeared in the March 1985 C Chest.) Because functions 11 and 12 force you to construct an FCB, and they search only the current directory, they're pretty hard to use.

Luckily, DOS versions 2 and higher provide two easier-to-use functions: "Find First" (0x4e) and "Find Next" (0x4f). When you pass Find First a pointer to a string holding a filename, it loads information about the requested file into the current disk transfer area (DTA). If you give an ambiguous file reference, subsequent files that match the reference are retrieved with a series of Find Next calls. Find Next must use the same DTA as Find First used.

If an error is encountered (such as not finding the requested file), the carry bit is set when the DOS interrupt returns, and an error code is put into the AX register. DOS v.2 sets the AX register to 0 on success, but DOS v.3 doesn't seem to do this; you actually have to look at the carry bit.

These two routines have several nice features. The name can contain ambiguous file references and directory specifiers, so you can search for a file anywhere in the file system. You can even use constructs like "." or ".\." if you want. Perhaps nicer, you can use / as well as \ to separate directory names in a complete file reference. (This is actually true for all the low-level MSDOS file functions. The insistence on that idiotic backslash is an anomaly of the MSDOS-supplied shell, command.com. Why couldn't they use / to separate directories and - for command line switches?) Another nice feature is that you can specify a disk id as part of the filename (c:/foo).

All is not wonderful, though. The two directory-searching routines do have problems. Unix directories are just files. The only difference between a Unix directory and any other file is the value of the directory attribute bit. MSDOS uses a similar mechanism, except that directories are special: they can't be accessed like other files, they have a length of zero, and so forth. However, they will show up in a directory search as if they were files (with an attribute bit that says you're looking at a directory).

So, if you request a directory from Find First, you'll get a single listing with the matching directory name. You won't get a listing of the files in that directory; to do this, you have to specify /*.* after the directory name. A search for "." will return a structure that actually contains the name of the parent directory.

A second problem is the root directory. MSDOS doesn't think that the root directory exists. If you request / or \ from Find First, it comes back with a file-not-found error. Similarly, requesting b: produces the same error. You need to ask for a file (or *.*).

Another problem is the volume label. A filename returned by DOS has all the blank padding removed and a period inserted between the name and extension. Unfortunately, DOS also puts the period into a volume label. If a disk is labeled LONGLABEL, the DOS directory search will yield LONGLABE.L as the volume label. Ls doesn't do anything to correct this

problem.

The final problem is actually a deficiency in the Lattice C I/O library. (The people at Lattice claim that the next revision of the compiler fixes this, but that doesn't help us now.) To use Find First/Next, you need to mess with the DTA (the place where the directory will end up after the

function call). Unfortunately, your disk I/O system uses the DTA. If you're not doing any file I/O, you can put the DTA where you want it (with a DOS function 0x1a call) and then forget about it. *But*, if you're working with files too, you'll have to put the DTA back where it came from to do file accesses.

C:\SRC\LS ls/

HARDFILE	<u>GAMES</u>	<u>SRC</u>	AUTOEXEC.BAT
<u>BIN</u>	<u>INCLUDE</u>	<u>TEXT</u>	COMMAND.COM
<u>DOS</u>	<u>LIB</u>	<u>UTIL</u>	CONFIG.SYS

3 files (22172 bytes, 21 K), 8 directories

C:\SRC\LS ls -la/

HARDFILE	0	12-20-84 17:22:18 LM
<u>BIN</u>	0	12-24-84 15:13:04 D
<u>DOS</u>	0	12-20-84 17:23:04 D
<u>GAMES</u>	0	12-24-84 15:20:32 D
<u>INCLUDE</u>	0	12-24-84 15:17:56 D
<u>LIB</u>	0	12-24-84 15:22:26 D
<u>SRC</u>	0	12-24-84 15:13:54 D
<u>TEXT</u>	0	12-24-84 15:20:58 D
<u>UTIL</u>	0	12-24-84 15:17:58 D
AUTOEXEC.BAT	95	4-09-85 17:42:50 M
COMMAND.COM	22042	8-14-84 8:00:00
CONFIG.SYS	35	1-22-85 22:19:06 M
IBMBIO.COM	8964	7-05-84 15:00:00 RHS
IBMDOS.COM	27920	7-05-84 15:00:00 RHS

3 files (59056 bytes, 58 K), 8 directories

C:\SRC\LS ls -a/text/drdobbs

.	<u>FOO</u>	LS.NR	SAVE.BAT
..	CROOT.NR	Q_BITMAP.NR	SORT.NR

5 files (56197 bytes, 54 K), 3 directories

C:\SRC\LS ls -c1fe/text/drdobbs

SAVE.BAT
CROOT.NR
LS.NR
Q_BITMAP.NR
SORT.NR

5 files (56197 bytes, 54 K)
C:\SRC\LS ls -d/text/drdobbs

FOO

1 directory

Figure 1.
Sample Outputs from Ls

DOS function 0x2f returns the current DTA in ES:BX. Unfortunately, the Lattice DOS interface functions (bdos, int86, etc.) won't let you get at the ES returned by the get DTA function. They push all the registers, do the DOS call, then restore the registers, overwriting the ES returned by DOS. Nor will a subsequent segread() call return the value you need for ES.

My solution to this problem was to write a more sensible DOS interface, which we'll talk about in a moment. Although ls doesn't need to put the

DTA back, an example of how to restore the DTA is given in Listing Five (page 41).

Program Description

ls itself appears in Listing One (page 25). MAXDIR (line 12) is the maximum number of files that will be listed; change this if you need more than 132 (132 names are 6 columns by 22 lines).

ls uses two subroutines that have appeared in previous columns: qsort (DDJ #102, April 1985) to sort the directory and getargs (DDJ #103,

May 1985) to parse command line arguments. The argument table for getargs is on lines 27-37 of Listing One (getargs.h on line 2 is used by getargs, as we discussed back in May).

The global variables on lines 19-26 are set by the various command line switches. The variables on lines 39-44 are more general purpose.

Directories are stored in an argv-like array of pointers to character strings: Dirs (line 39). Dirv and Dirc (40-41) operate like argv and argc. Total (line 42) is the total size, in bytes, of all listed files. Numfiles and Numdirs (43-44) are the number of files and directories found, respectively.

Find_first() and find_next() (lines 47-77) do the DOS calls. They use the DOS interface function given in Listing Three (page 34).

The REGS structure is *not* the structure of the same name defined in the dos.h supplied by Lattice. This version of REGS is defined in mydos.h (Listing Two, page 33, lines 14-23). REGS is actually a functional superset of the Lattice REGS structure (that is, you can use the REGS defined in mydos.h to talk to the Lattice interface functions, but you can't use the Lattice REGS to talk to the dos() function in Listing Three).

REGS has fields for all the registers you need to access MSDOS, including the segment registers. It is a union that defines the normal registers so that you can easily access the high, low, or both bytes. The two register images are superimposed (i.e., definitions for byte offsets are superimposed over definitions for word offsets). For example, given a pointer p to a REGS structure, p->x.ax accesses the AX register, p->h.ah gets the high byte (the AH register), and p->h.al gets the low byte (the AL register). You can retrieve only the AX, BX, CX, and DX registers in this way. The others (SI, DI, ES, CS, SS, DS) are all accessed through the x part of the union (p->x.es).

The interface function, dos() (Listing Three), is called with dos(regp); where regp is a pointer to a REGS type structure. It returns the status register, as returned by DOS. You can use dos() only with the

*C Programmers
Quit Working
So Hard!*



THE GREENLEAF FUNCTIONS™

The GREENLEAF FUNCTIONS GENERAL LIBRARY

Strength in DOS, video, string, printer, async, and systems interface. All DOS 1 and 2 functions are in assembler for speed.

All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library**.) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions**. Simply the finest C library (and the most extensive). All ready for you.

THE GREENLEAF FUNCTIONS™

The Library of C Functions that probably has just what you need... **TODAY!**

- already has what you're working to re-invent
- already has over 200 functions for the IBM PC, XT, AT, and compatibles
- already complete... already tested... on the shelf
- already has demo programs and source code
- already compatible with all popular compilers
- already supports all memory models, DOS 1.1, 2.0, 2.1
- already optimized (parts in assembler) for speed and density
- already in use by thousands of customers worldwide
- already available from stock (your dealer probably has it)
- It's called the **GREENLEAF FUNCTIONS**.

The Library of C Functions is Waiting for You

Specify compiler when ordering. Add \$7.00 for UPS second-day air (or \$5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check or P.O. In stock, shipped same day.

- | | | |
|---------------------|-------|-------------------------------|
| ■ General Libraries | \$185 | For Information: 214-446-8641 |
| ■ Comm Library | \$185 | |
| ■ C/C86 Compiler | \$349 | |
| ■ Lattice C | \$395 | |
| ■ Mark Williams | \$475 | |

Prices are subject to change without notice.



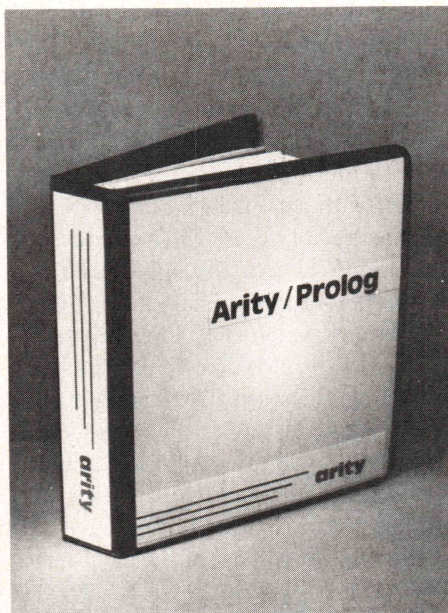
2101 HICKORY DR.
CARROLLTON, TX 75006

Circle no. 43 on reader service card.

More Power Than You Thought Possible

Arity offers the first serious implementation of Prolog for IBM personal computers. Arity/Prolog is a powerful, highly optimized, and extended version of the logic programming language Prolog. Imagine building software applications with a language that solves problems through deduction and logical inference. The task of creating complex programs is much faster and easier, resulting in lower development costs. Arity/Prolog is now in use in a wide range of applications in industry, business, research, and education. The solution—the *Arity/Prolog Interpreter*:

- Source level debugger
- Virtual databases, each with a workspace of 16 megabytes
- Floating-point arithmetic
- String support for efficient text handling



- Interface to assembly language and 'C'
- Text screen manipulation
- Integrated programming shell to MSDOS
- Comprehensive set of evaluable predicates
- Definite clause grammar support

Arity/Prolog Interpreter \$495.00

Arity also offers the *Arity/Prolog Compiler and Interpreter*, a sophisticated development environment for building AI applications. Essential for producing fast, serious production code.

Arity/Prolog Compiler and Interpreter \$1950.00

The *Arity/Prolog Demo Disk* is available for \$19.95. ■ Arity/Prolog products run on the IBM PC, XT, AT, and all IBM compatibles. ■ To order, call (617) 371-2422 or use the order form below.

arity corporation 358 Baker Avenue, Concord, MA 01742

Name _____
 Organization _____
 Address _____

☐ Enclosed is a check or money order to Arity Corporation

☐ Please bill my
☐ Mastercard ☐ Visa ☐ American Express

Account #

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Valid ____/____/____ signature _____

Quantity	Product	Unit Price	Total Price
	Arity/Prolog Compiler & Interpreter	\$1950.00	
	Arity/Prolog Interpreter	\$ 495.00	
	Arity/Prolog Demo Disk	\$ 19.95	
Subtotal			
MA residents add 5% sales tax			
Total Amount			

☐ Please send me more information about Arity and Arity/Prolog

arity 358 Baker Avenue, Concord, MA 01742

M-AD-0

LARGE DISCOUNTS FROM JOHN D. OWENS ASSOCIATES

MACROTECH MI-286: 80286/Z-80H DUAL PROCESSOR S-100 CPU BOARD: ... **\$876** MSR RAM: 120NS, high-speed dynamic RAM, Works with CompuPro 8085/8088; MI-286 and others: 256K .. **\$556** 512K ... **\$716** STATIC RAM: Substitute for RAM 22 and RAM 23: 256K STATIC: **\$960** 512K STATIC: **\$1,800**

EMERALD SYSTEMS HARD DISK and TAPE SUBSYSTEMS
High capacity! Up to 280 MB! Emerald has overcome the 32MB DOS limitation! Multiple volumes per physical drive. Back up and restore utilities.

HOUSTON INSTRUMENTS: Plotters: DMP 41 OR 42: .. **\$2,397**, DMP 29: ... **\$1,838** DIGITIZERS: DT11 .. **\$694**; DT114 **\$750**; DT11AA **\$714**
New! 14 pen DMP 51 and 52: ... **\$4,796**

ILLUMINATED TECHNOLOGY
S-100 COLOR GRAPHICS: **\$1,116**

NEC APC III: 80816 MS-DOS system w/ spectacular graphics: 20% off list

SEMEDISK 2MB DISK EMULATOR FOR IBM PC and EPSON QX 10: **\$2,050**
S-100IMB: .. **\$1,472** S-1002MB: .. **\$2,091**

LOMAS: 80286, CCP/M and MS-DOS S-100 Systems with COLOR GRAPHICS for IBM-PC compatibility.

RACTER: Interactive conversational software for IBM-PC. Jan 1985 Scientific American called RACTER, "... extremely funny." RACTER is light years ahead of Eliza! **\$69.95.**

MIST + CONEXUS: Integrated software for IBM-PC combining telecommunications, text processing, data base management.

BIG DISCOUNTS on COMPUPRO, IMS, BYAD, many others. Prices & availability subject to change without notice.

WRITE OR CALL FOR PRODUCT LITERATURE.

WE EXPORT: Overseas Callers: TWX 710 588 2844 (OWENSASSOC NYK). EASY LINK MAILBOX ADDRESS: 62840 768.

ORDER BY PHONE, LETTER, TELEX OR EASY LINK. We accept VISA and Mastercard. Shipping \$5 per board in continental USA.

**JOHN D. OWENS
ASSOCIATES**

12 SCHUBERT STREET
STATEN ISLAND, N.Y. 10305
(718) 448 6283 (718) 448 6298
(718) 448 2913

JOHN D. OWENS ASSOCIATES JOHN D. OWENS

small and one of the medium (the P) memory models because it assumes a 16-bit pointer.

Dos() copies all fields of *regp except x.ss and x.cs into their equivalent registers (lines 59-66 of Listing Three). DOS is invoked with the INT 21 on line 68, then *all* the registers are copied back into *regp. The routine returns with registers restored to their original state (i.e., before the dos() call). The function gregs() (lines 93-124) gets the register contents but doesn't do a DOS call; it's used for initializing a REGS structure.

Find_first() and find_next() both return 0 on success or the error code passed back by DOS on failure. When they succeed, the current DTA will be loaded with a shuffled-around version of the directory entry for the requested file. You use the FILE_INFO structure, typedefed in mydos.h (Listing Two, lines 30-39), to access this information. Mydos.h also has a bunch of macros for extracting stuff from packed fields (lines 59-79). With these, you can get at the various fields of the creation time and date, as well as test for specific attributes.

Dirtoa() (Listing One, lines 86-135) converts a FILE_INFO structure into an ASCII string (see the figure). The ANSI escape sequences used for underlining and boldfacing are added to the string in dirtoa().

The final directory-related routine is fixup_name() (Listing One, lines 174-223). This routine takes care of all the problems with accessing the root directory and getting the files inside a specified directory that we discussed earlier. It appends *.* or /*.* onto directory names where necessary.

Add_entry() (Listing One, lines 225-253) puts the strings returned from dirtoa() into the Dirv array. It also processes the -a and -f flags. If an attribute bit is set when you call find_first() or find_next(), then only those files that have the indicated attributes will be found; the -d option is processed on line 281 using this mechanism. Unfortunately, because there's no way to request files (only directories), -f must be processed by hand in add_entry() (lines 241-246).

The only remaining nonobvious function in ls.c is ecmp() (lines 147-162), which is used by qsort to do a sort by extension. It skips over to the extension parts of the two strings, compares the extensions, and if they match, goes back and compares the actual filename. Refer to the April C Chest for more information on how qsort works.

Multicolumn Printing

The multicolumn printing is done by ptext() (Listing Four, page 37), which prints out an argv-like array of pointers to strings in multicolumn format. Ptext() originally was written for a version of the Unix print utility pr and works quite well in that application.

Ptext()'s calling syntax is:

```
ptext(linec, linev, outfile, numcols,
      colwidth, numrows);
char **linev;
FILE *outfile;
```

Linev is the array of string pointers, linec is the number of pointers in the array, outfile is the stream to which output will be sent, numcols and numrows are the number of rows and columns, respectively, and colwidth is the width of a single column.

Ptext() is called in Listing One on line 170. Here num_cols is the number of columns. The column width is 80/Num_cols. The number of rows is computed with (dirc/Num_cols) + (dirc % Num_cols != 0); dirc is the total number of names to be printed. Dirc % Num_cols != 0 evaluates to 1 if Num_cols doesn't divide evenly into dirc and to 0 if it does divide evenly.

Several practical problems having to do with multicolumn printing will vary from application to application. The first problem is a lone carriage return. Some text formatters underline text by printing the text itself, then a carriage return, then a line containing all the underscores. When you have several columns, you can't print the carriage return because that takes you to the left edge of the page rather than the left edge of the current column. The problem is solved in ptext() by backspacing to the correct

place. If your printer doesn't support a backspace, you'll have to modify this part of the code (lines 93-103).

The next problem is ESC sequences (which take up no space in the output). Different sequences are composed of different numbers of characters. Rather than try to recognize various escape sequences, ptext() just assumes that all sequences use four characters (one for the ESC and three normal characters that follow). This assumption lets us handle the various Set Graphics Rendition (SGR) sequences correctly (we need these for the underlining and boldfacing), but it won't handle the escape sequences needed to change fonts on printers and the like. So, you may need to make this part of ptext() more sophisticated (the relevant code is on lines 117-129).

The third problem is with the ANSI.SYS driver provided by IBM (at least with DOS v.3 running on a PC/AT—I'd appreciate someone testing other versions of DOS on other ma-

chines and sending me the results). If you try to underline the left-most character on a line, the entire line ends up underlined. If you explicitly turn off underlining, the nonunderlined characters overwrite the underscores, but underscores still extend from the last character you wrote to the physical end of line on the screen (i.e., from the place where you wrote a carriage return to the right edge of the screen). To make matters worse, the underline attribute sticks on, so subsequent lines end up underlined too. I didn't want to rewrite ANSI.SYS, so I kludged a solution to this problem in ptext(). If "IBM" is #defined at the top of the module, a space will be inserted as the left-most character of every line. I realize this isn't a real solution, but it works everywhere I've needed to use ptext().

While we're on the subject of ANSI.SYS, I found another bug a few days ago. The "Erase in Display" and "Erase in Line" leave the cursor in the wrong place. Erase in Display

should home the cursor; instead, it leaves the cursor at the left edge of row 2. Similarly, Erase in Line puts the cursor at the left edge of the next line; it shouldn't move the cursor at all. Obviously, the driver is just printing blanks to erase the line, but it should put the cursor back where it belongs when it's done.

The Erase in Line problem is particularly bad when you try to erase the bottom line: the screen scrolls up a line when the cursor is sent past the bottom, so, even if you put the cursor back where it belongs, you lose the top line of the screen. If anyone knows any other bugs in ANSI.SYS (or has fixes for them, maybe a new version of ANSI.SYS), please send them in so I can put them in the column.

This month we talked about directories; next month we'll continue the discussion with a version of the Unix utility "make," which automates the recompilation of modular programs.

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 191.

C Chest (Text begins on page 20)

Listing One

```

1: #include <stdio.h>
2: #include <getargs.h>
3: #include <mydos.h>

4: /*-----
5: *                LS.C: An MSDOS directory utility
6: *
7: *      (c) Copyright 1985, Allen I. Holub. All rights reserved.
8: *      This program may be reproduced for personal, non-profit use only.
9: *-----
10: */

11: #define CARRY    0x01    /* Carry flag mask for an 8086 */
12: #define MAXDIR   132    /* Largest directory which will be printed */

13: #define BOLDFACE  "\033[1m" /* Ansi esc sequence to turn bold face on */
14: #define UNDERLINE "\033[4m" /*                "                underline on */
15: #define ALL_OFF   "\033[0m" /*                "                attributes off */
16:
17: #define EOS(s)    while(*s) s++ /* Position s at the end of the string */

18: /*-----

19: int      Longfmt      = 0;    /* Global variables set by command line */
20: int      Unsorted     = 0;    /* switches. */
21: int      Files_only   = 0;
22: int      No_graphics  = 0;
23: int      Dirs_only    = 0;
24: int      List_all     = 0;
25: int      Num_cols     = 5;
26: int      Ext_sort     = 0;

```

(Continued on next page)

Listing One

```

27: ARG Argtab[] =
28: {
29:     { 'a', BOOLEAN, &List_all, "List all files (including hidden)" },
30:     { 'c', INTEGER, &Num_cols, "Print output in <num> columns" },
31:     { 'd', BOOLEAN, &Dirs_only, "List only directories" },
32:     { 'e', BOOLEAN, &Ext_sort, "Sort by extension" },
33:     { 'f', BOOLEAN, &Files_only, "List only files" },
34:     { 'l', BOOLEAN, &Longfmt, "Print directory in long format" },
35:     { 's', BOOLEAN, &No_graphics, "Suppress ANSI graphics chars" },
36:     { 'u', BOOLEAN, &Unsorted, "Print directory unsorted" },
37: };

38: #define TSIZE (sizeof(Argtab)/sizeof(ARG))

39: static char *Dirs[MAXDIR] ; /* Place to put directory */
40: static char **Dirv = Dirs ; /* Like argv for directory */
41: static int Dirc = 0 ; /* Like argc for directory */
42: static long Total = 0L ; /* Sum of sizes of all files */
43: static int Numfiles = 0 ; /* Number of files found */
44: static int Numdirs = 0 ; /* " " directories " */
45: extern char *malloc();

46: /*-----*/

47: find_first( filespec, attributes, regp )
48: char *filespec ;
49: short attributes;
50: REGS *regp;
51: {
52:     /* Get directory information for the indicated file.
53:      * Ambiguous file references are ok but you have to use
54:      * find_next to get the rest of the file references.
55:      * In this case, The regs structure used by find_first
56:      * must be passed to find_next. 0 is returned on success,
57:      * otherwise the DOS error code is returned.
58:     */

59:     regp->h.ah = (char) FINDFIRST ;
60:     regp->x.dx = (short) filespec ;
61:     regp->x.cx = attributes ;

62:     return (int)( (dos(regp) & CARRY) ? regp->x.ax : 0 );
63: }

64: /*-----*/

65: find_next ( regp )
66: REGS *regp;
67: {
68:     /* Get the next file in an ambiguous file reference. A
69:      * call to this function must be preceded by a
70:      * find_first call. The regp argument must be the
71:      * same register image used by the find_first call.
72:      * 0 is returned on success, otherwise the error code
73:      * generated by DOS is returned.
74:     */

75:     regp->h.ah = FINDNEXT ;
76:     return (int)( (dos(regp) & CARRY) ? regp->x.ax : 0 );
77: }

78: /*-----*/

79: doscall( id, regp ) /* Do the DOS system call */
80: REGS *regp; /* specified by "id", *regp */
81: { /* is modified to reflect the */
82:     regp->h.ah = id ; /* results of that call */

```



```

83:         dos( regp );
84: }

85: /*-----*/
86: dirtoa( s, p, longfmt )
87: char      *s;
88: FILE_INFO *p;
89: {
90:     /* Convert a FILE_INFO structure to an ascii string. In longfmt
91:      * all information about the file (name, size, date & time,
92:      * mode) is used. Possible modes are:
93:      *
94:      *      R - Read only          H - Hidden
95:      *      S - System             L - Volume label
96:      *      D - Directory
97:      *      M - archive bit is set (file has been modified)
98:      *
99:      * If longfmt isn't true, only the name is printed. Maximum
100:     * string length in long format is 52, in short format is 21
101:     * Volume label names are printed in bold face (since this will
102:     * put an ESC in the name the volume label will sort to the front
103:     * of the list). Directories are printed underlined (these will
104:     * immediately follow the directories). The number of characters
105:     * in the string (not including the terminating null) is returned.
106:     */

107:     char *startstr = s;
108:     int i;

109:     if( !No_graphics && (IS_LABEL(p) || IS_SUBDIR(p)) )
110:         sprintf(s, "%s%s", IS_LABEL(p) ? BOLDFACE : UNDERLINE,
111:                 p->fi_name, ALL_OFF );
112:     else
113:         sprintf(s, "%s", p->fi_name );

114:     EOS(s);

115:     if( longfmt )
116:     {
117:         for( i = strlen(p->fi_name); i++ < 12 ; *s++ = ' ' )
118:             ; /* Pad out the name field */

119:         sprintf( s, " %6ld ", p->fi_fsize );

120:         EOS(s) ;
121:         sprintf(s, "%2d-%02d-%02d %2d:%02d:%02d",
122:                 C_MONTH(p), C_DAY(p), C_YEAR(p)-1900,
123:                 C_HR(p), C_MIN(p), C_SEC(p) );
124:         s += 17 ;
125:         *s++ = ' ';

126:         if( IS_READONLY(p) ) *s++ = 'R' ;
127:         if( IS_HIDDEN(p) ) *s++ = 'H' ;
128:         if( IS_SYSTEM(p) ) *s++ = 'S' ;
129:         if( IS_LABEL(p) ) *s++ = 'L' ;
130:         if( IS_SUBDIR(p) ) *s++ = 'D' ;
131:         if( IS_DIRTY(p) ) *s++ = 'M' ;
132:         *s = 0;
133:     }
134:     return( s - startstr );
135: }

136: /*-----*/

137: haswild(s)
138: char      *s;
139: {
140:     /*      Return true if s has a '*' or '?' in it      */

141:     for( ; *s ; s++)
142:         if( *s == '*' || *s == '?' )
143:             return 1;

```

(Continued on next page)

C Chest

Listing One

(Listing Continued, text begins on page 20)

```
144:         return 0;
145: }

146: /*-----*/

147: ecmp( slp, s2p )
148: char  **slp, **s2p;
149: {
150:     /* Comparison routine for sorting a directory by extension */
151:     char  *s1, *s2;
152:     int    rval;
153:     for( s1 = *slp; *s1 && *s1 != '.' && *s1 != ' '; s1++ )
154:         ;
155:     for( s2 = *s2p; *s2 && *s2 != '.' && *s2 != ' '; s2++ )
156:         ;
157:     if( rval = strcmp(s1, s2) ) /* If the extensions don't match */
158:         return rval;          /* return the strcmp value */
159:     for(s1=*slp, s2=*s2p ; *s1==*s2 && *s1 && *s1 != '.'; s1++, s2++)
160:         ;
161:     return( (*s1 == '.' ? 0 : *s1) - (*s2 == '.' ? 0 : *s2) );
162: }

163: /*-----*/

164: printdir(dirc, dirv)
165: int      dirc ;
166: char     **dirv ;
167: {
168:     if( !Unsorted )
169:         qsort( dirv, dirc, sizeof(*dirv), Ext_sort ? &ecmp : 0);
170:     ptext( dirc, dirv, stdout, Num_cols, 80/Num_cols,
171:           (dirc/Num_cols) + (dirc % Num_cols != 0) );
172: }

173: /*-----*/

174: char  *fixup_name( name, regs, info )
175: char  *name;
176: REGS  *regs;
177: FILE_INFO *info;
178: {
179:     /* If the name specifies an implicit file (ie. it asks for
180:      * the directory rather than the files in the directory),
181:      * modify it to ask for files (eg. ".." becomes "..\*..*").
182:      */
183:     static char      buf[80], *p ;
184:     if( !find_first( name, ALL, regs ) )
185:     {
186:         /* If the requested name was found, and that name is
187:          * a sub directory, append \*..* to the requested name
188:          */
189:         if( !IS_SUBDIR(info) )
190:             return( name );
191:         strncpy( buf, name, 80 );
192:         strncat( buf, "\\*..*", 80 );
193:         return ( buf );
194:     }
195:     else
```



```

196: {
197:     /* If you didn't find anything, and a non-explicit
198:     * directory is requested (ie. "..\.." or "b:") then
199:     * set up the name to get the root directory on the
200:     * indicated disk.
201:     */

202:     if( name[1] == ':' ) /* Take care of the */
203:     { /* disk designator first */
204:         buf[0] = name[0];
205:         buf[1] = ':' ;

206:         if( !(name += 2) )
207:         {
208:             buf[2] = '*'; /* If the current dir */
209:             buf[3] = '.'; /* on another disk is */
210:             buf[4] = '*'; /* requested... */
211:             buf[5] = 0 ;
212:             return( buf );
213:         }
214:         else
215:             buf[2] = 0 ;
216:     }

217:     for( p = name ; *p ; p++ )
218:         if( !( *p == '.' || *p == '\\ ' || *p == '/' ) )
219:             return( name );

220:     strncat( buf, "\\*.*", 80 );
221:     return( buf );
222: }
223: }

224: /*-----*/

225: add_entry( p )
226: FILE_INFO *p;
227: {
228:     /* Add an entry to the directory array (dirv) and update
229:     * various associated globals. Hidden files are not
230:     * printed unless -a was given on the command line.
231:     * (which will cause List_all to be set). As with
232:     * Unix, file names starting with the character '.' are
233:     * also not printed unless -a is specified. If -d was
234:     * given on the command line (Dirs_only will be true)
235:     * and files won't be printed.
236:     */

237:     char buf[64];
238:     register int strlen;

239:     if( !List_all && (IS_HIDDEN(p) || *p->fi_name == '.') )
240:         return 1;

241:     if( IS_SUBDIR(p) ) /* Entry is a directory */
242:         Numdirs++; /* Adjust directory count */

243:     else if( Dirs_only ) /* Entry is a file */
244:         return 1; /* Don't print files */

245:     else if( !IS_LABEL(p) ) /* Entry is a file */
246:         Numfiles++; /* Adjust file count */

247:     strlen = dirtoa( buf, p, Longfmt );
248:     if( ++Dirc > MAXDIR || !( *Dirv = malloc( strlen + 1 ) ) )
249:         return 0;

250:     strcpy( *Dirv++, buf );
251:     Total += p->fi_fsize;
252:     return 1;
253: }

```

(Continued on next page)

The word is out. dBASE III is the world

"dBASE III is the most mature, best performing business programming language available on any computer."

Adam Green
Personal Software
August, 1984

"With its impressive specifications, dBASE III is clearly designed for the future. With 128 fields per record, a user could feasibly have access to 1,280 fields of information, opening the micro to applications heretofore reserved for large systems."

Edward Bride
Software News
July, 1984

"Ashton-Tate has created dBASE III, the connoisseur's database management system."

Russell Lipton
Business Computer Systems
October, 1984

"Enter dBASE III. Not only is this whiz of a data base loaded with features customers asked for, it sports an enhanced language that brings database programming into a new dimension."

Daryl Rubin
Computer Language
January, 1985

"dBASE III makes that wish list a reality..."

Marc Stern
InfoWorld
October 15, 1984

"Ashton-Tate has done it again. dBASE III should assure the company an extended stay at the top of the microcomputer database management system market."

Rowland Archer
Business Computer Systems
November, 1984

"It's a classy product. It's easy to use, has incredible power, polish and finesse. It's an easy program for the novice. There is untapped power for the experienced user."

Robert Byers
Data Based Advisor
September, 1984



's best data management software.

dBASE III™ is the industry standard for managing information on personal computers. Written to take full advantage of the capabilities of the IBM® PC, Personal Computer AT™, compatible computers and other popular personal computers, dBASE III offers virtually limitless data-handling capabilities. And, for novices, a built-in

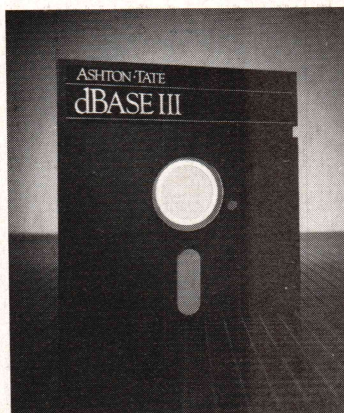
Assistant feature makes dBASE III the data management system of choice.

With so many great reviews, dBASE III is the clear choice in data management.

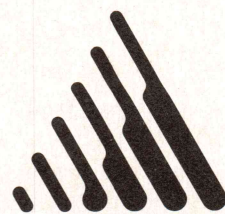
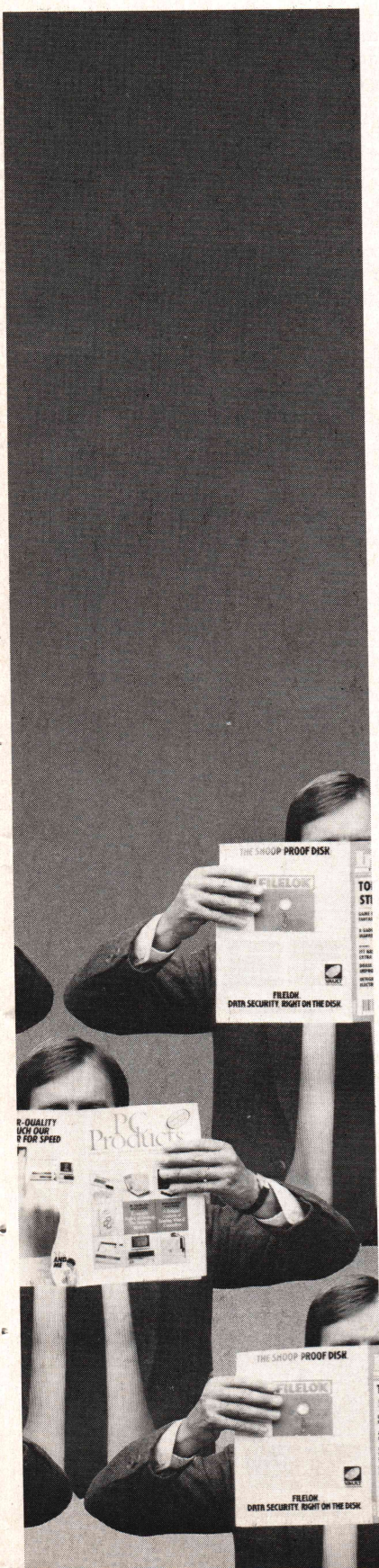
For a dealer near you call (800) 437-4329, ext. 2333. In Colorado call

(303) 799-4900, ext. 2333.

dBASE III and Ashton-Tate are trademarks of Ashton-Tate. IBM is a registered trademark and Personal Computer AT is a trademark of International Business Machines Corp.



dBASE III. The data management standard.



ASHTON-TATE
S O F T W A R E

Circle no. 5 on reader service card.

C Chest

(Listing Continued, text begins on page 20)

Listing One

```

254: /*-----*/
255: main(argc, argv)
256: int     argc;
257: char    **argv;
258: {
259:     /* Get a directory. Given a list of files on the command line
260:      * (file names may be ambiguous) will print a sorted directory
261:      * of those files along with a total file and byte count.
262:      */

263:     static REGS      regs          ;      /* Needed for DOS calls */
264:     static FILE_INFO info          ;      /* DOS puts dirs here  */
265:     static char      *name = " *.*";      /* File name           */

266:     gregs ( &regs );                      /* Initialize the regs structures */
267:     regs.x.dx = (word) &info;              /* Change the Disk Transfer Addr */
268:     doscall( SETDTA, &regs );              /* to point at info structure    */

269:     argc = getargs( argc, argv, Argstab, TSIZE );

270:     if( Longfmt )                          /* -l is always printed in 1 column */
271:         Num_cols = 1;                      /* even if -cN was given on the cmd */
272:                                             /* line                               */

273:     if( argc >= 2 )                        /* Put the requested file name into */
274:     {                                      /* "name." The default is *.*       */
275:         name = *(++argv);
276:         if( argc == 2 && !haswild(name) )
277:             name = fixup_name( name, &regs, &info );
278:     }

279:     do
280:     {
281:         if( !find_first(name, Files_only ? ALL_FILES : ALL, &regs))
282:         {
283:             if( !add_entry(&info) )
284:                 break;

285:             if( haswild(name) )
286:                 while( !find_next( &regs ) )
287:                     if( !add_entry(&info) )
288:                         goto abort;
289:         }
290:         name = *(++argv);

291:     } while( --argc > 1 );

292: abort: printf ( "\n");
293:         printdir ( Dirc, Dirs );
294:         printf ( "\n");
295:         if( Numfiles )
296:         {
297:             printf("%d file%s (%ld bytes, %d K)",
298:                   Numfiles, Numfiles == 1 ? "" : "s",
299:                   Total, Total/1024 );
300:         }

301:         if( Numdirs )
302:         {
303:             if( Numfiles )
304:                 printf(", ");

305:             printf("%d director%s", Numdirs, Numdirs ==1 ? "y" : "ies");
306:         }
307: }

```

End Listing One

Listing Two

```

1: /*-----+
2: *
3: *      MYDOS.H      Various defines for talking to dos
4: *
5: *-----+
6: *
7: *      Typdefs for using dos(). Note that this structure can be used
8: *      by the various routines supplied by Lattice (intdos, bdos etc.)
9: *      but dos() can't use the structure defined in Lattice's dos.h.
10: *
11: *      Since "REGS" and "byte" are both also defined in dos.h, you
12: *      shouldn't #include both of them in the same place.
13: */

14: typedef short      word;
15: typedef char       byte;

16: struct LREG { word  ax,      bx,      cx,      dx,      si, di, es, cs, ss, ds;};
17: struct SREG { byte  al,ah, bl,bh, cl,ch, dl,dh;      };

18: typedef union
19: {
20:     struct LREG      x ;
21:     struct SREG      h ;
22: }
23: REGS;

24: /*-----+
25: *      Stuff needed to get a directory from MSDOS
26: *
27: *      The FILE_INFO type structure is filled by a find first or find
28: *      next command (dos system calls 0x4e and 0x4f).
29: */

30: typedef struct
31: {
32:     char      fi_resv[21];      /* Bytes 0-20   Reserved by DOS      */
33:     char      fi_attr;         /* Byte  21    File attribute      */
34:     short     fi_time;         /* Bytes 22-23 Create/update time   */
35:     short     fi_date;         /* Bytes 24-25 Create/update date   */
36:     long      fi_fsize;        /* Bytes 26-27 File size in bytes   */
37:     char      fi_name[13];     /* Bytes 28-40 File name & extension */
38: }
39: FILE_INFO;

40: /*
41: *      Macros to extract information from a FILE_INFO. In all these macros
42: *      the argument "p" is a pointer to a FILE_INFO structure. Note that
43: *      the C_YEAR and C_SEC macros compensate for MSDOS weirdnesses.
44: *
45: *      IS_READONLY(p)      File is read only.
46: *      IS_HIDDEN(p)       File is invisible in normal directory searches
47: *      IS_SYSTEM(p)       File is a system file
48: *      IS_LABEL(p)        Info is a volume label, not a file.
49: *      IS_SUBDIR(p)       File is a directory
50: *      IS_DIRTY(p)        True when file is written to and closed, set to
51: *                          False by the program backup.
52: *      C_HR(p)            Hour of last update or create   (0-23)
53: *      C_MIN(p)           Minute of last update or create (0-59)
54: *      C_SEC(p)           Second of last update or create (0-59)
55: *      C_YEAR(p)          Year of last update or create   (1980-2099)
56: *      C_MONTH(p)         Month of last update or create  (1-12)
57: *      C_DAY(p)           Day of last update or create    (1-31)
58: */

59: #define READONLY      0x01      /* Attribute bits      */
60: #define HIDDEN        0x02
61: #define SYSTEM        0x04

```

(Continued on next page)

Listing Two

```
62: #define LABEL          0x08
63: #define SUBDIR          0x10
64: #define DIRTY           0x20

65: #define ALL              (READONLY | DIRTY | SYSTEM | HIDDEN | SUBDIR | LABEL)
66: #define ALL_FILES        (READONLY | DIRTY | SYSTEM | HIDDEN )
67: #define NORM_FILES        (READONLY | DIRTY )

68: #define IS_READONLY(p)    ((p)->fi_attrib & READONLY )
69: #define IS_HIDDEN(p)      ((p)->fi_attrib & HIDDEN )
70: #define IS_SYSTEM(p)      ((p)->fi_attrib & SYSTEM )
71: #define IS_LABEL(p)       ((p)->fi_attrib & LABEL )
72: #define IS_SUBDIR(p)      ((p)->fi_attrib & SUBDIR )
73: #define IS_DIRTY(p)       ((p)->fi_attrib & DIRTY )

74: #define C_HR(p)           ( ((p)->fi_time >> 11) & 0x1f )
75: #define C_MIN(p)          ( ((p)->fi_time >> 5) & 0x3f )
76: #define C_SEC(p)          ( ((p)->fi_time << 1) & 0x3e )
77: #define C_YEAR(p)         (((p)->fi_date >> 9) & 0x7f ) + 1980)
78: #define C_MONTH(p)        ( ((p)->fi_date >> 5) & 0x0f )
79: #define C_DAY(p)          ( ((p)->fi_date ) & 0x1f )

80: /*-----
81:  *      Directory related BDOS function numbers
82:  */

83: #define FINDFIRST         0x4e
84: #define FINDNEXT          0x4f
85: #define SETDTA            0x1a
86: #define GETDTA            0x2f
```

End Listing Two

Listing Three

```
1:      TITLE      DOS INTERFACE FUNCTION
2:      SUBTTL     Copyright 1985 by Allen I. Holub
3:      NAME       DOS
4:      INCLUDE    DOS.MAC

5: ;-----
6: ;
7: ; Data types: The REGS structure is defined as type allregs in mydos.h
8: ;

9: REGS      STRUC          ; Structure used to transfer register
10: AX_REG    DW            0 ; contents. Note that this structure is
11: BX_REG    DW            0 ; a superset of that defined in the
12: CX_REG    DW            0 ; dos.h supplied by Lattice. This particular
13: DX_REG    DW            0 ; structure is defined in \include\mydos.h
14: SI_REG    DW            0
15: DI_REG    DW            0
16: ES_REG    DW            0
17: CS_REG    DW            0
18: SS_REG    DW            0
19: DS_REG    DW            0
20: REGS      ENDS

21: OFF      EQU          4      ; Offset to arguments

22: ;-----
23: ;
24: ; name:          dos  --  Do a dos function call
25: ;
26: ; synopsis:      #include <mydos.h>
27: ;
```

(Continued on page 36)

THE PROGRAMMER'S SHOP™

helps save time, money and cut frustrations. Compare, evaluate, and find products.

SERVICES

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- Over 700 products
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 700 products

Free Literature - Compare Products

Evaluate products Compare competitors. Learn about new alternatives. One free call brings information on just about any programming need. Ask for any "Packet" or "Addon Packet" □ ADA, Modula □ "AI" □ BASIC □ "C" □ COBOL □ Editors □ FORTH □ FORTRAN □ PASCAL □ UNIX/PC or □ Debuggers, Linkers, etc.

RECENT DISCOVERIES

SMALL TALK for PC DOS - "Methods" has objects, windows, browser, inspector. PCDOS \$239

ARTIFICIAL INTELLIGENCE

ExpertLISP - Interpreter: Common LISP syntax, lexical scoping, toolbox, graphics. Native code COMPILER. 512K MAC \$465

ExpertEASE - Expert system tool. Develop by describing examples of how you decide. PCDOS \$625

EXSYS - Expert System building tool. Full RAM, Probability, Why, serious, files PCDOS \$275

GC LISP - "COMMON LISP", Help, tutorial, co-routines, compiled functions, thorough. PCDOS Call

INSIGHT 1 - Expert Sys. Dev't, decent PCDOS \$ 95

M Prolog - full, rich, separate work spaces. MSDOS \$725

PROLOG-86 - Learn fast, Standard, tutorials, samples of Natural Language, Exp. Sys. MSDOS \$125

TLC LISP - "LISP-machine"-like, all RAM, classes, turtle graphics 8087. CP/M-86, MSDOS \$235

EDITORS FOR PROGRAMMING

BRIEF Programmer's Editor - undo, windows, reconfigurable, macro programs, powerful. PCDOS \$195

VEDIT - well liked, macros, buffers, CPM-80-86, MSDOS, PCDOS \$119

MACINTOSH

We evaluate, carry every available programmers product. Ask for a packet describing OVER 20 PRODUCTS

C LANGUAGE

C Terp Interpreter by Gimbel, full K&R, .OBJ and ASM interface, 8087. MSDOS \$275

INSTANT C - Interactive development - Edit, Source Debug, run. Edit to Run - 3 Secs. MSDOS \$445

"INTRODUCING C" - Interactive C to learn fast. 500 page tutorial, examples, graphics. PCDOS \$ 95

MEGAMAX C - native Macintosh has fast compile, tight code, K&R, toolkit, .OBJ, DisASM MAC \$275

Wizard C - Lattice C compatible, full sys. III syntax, lint included, fast, lib. source. MSDOS \$450

C ADDONS

APPLICATION TOOLKIT by Shaw - Complete: ISAM, Screen, Overlay mgmt, report gen, Strings, String math. Source. CPM, MSDOS \$475

COMMUNICATIONS by Greenleaf (\$159) or Software horizons (\$139) includes Modem7, interrupts, etc. Source. Ask for Greenleaf demo.

C SHARP Realtime Toolkit - well supported, thorough, portable, objects, state sys. Source MANY \$600

C Index + - full B+Tree, variable length fields. Source, no royalties. MSDOS \$369

PC/LINT - Small, big model. Batch option. Lattice, C86 MSDOS \$ 95

FORTRAN LANGUAGE

MacFORTRAN - full '77, '66 option, toolbox, debugger, 128K or 512K, ASM-out option MAC \$375

RM/Fortran - Full '77, BIG ARRAYS, 8087, optimize, back trace, debug. MSDOS \$525

Ask about Microsoft, Supersoft, others.

OTHER LANGUAGES

ASSEMBLER - ask about FASM-86 (\$95), ED/ASM (\$95) - both are fast, compatible, or MASM (\$125), improvements.

BetterBASIC all RAM, modules, structure. BASICA-like PCDOS \$185

SNOBOL 4 + - great for strings, patterns. CPM86, MSDOS \$ 85

SUPPORT PRODUCTS

BASIC DEVELOPMENT SYSTEM - (BDS) for BASICA; Adds Renum, crossref, compress. PCDOS \$115

CODESIFTER - Execution PRO-FILER. Spot bottlenecks. Symbolic, automatic. PCDOS \$109

FASTER C - Lattice users eliminate Link step. Normal 27 seconds. Faster C in 13 sec. MSDOS \$ 95

PLINK-86 for Overlays, most lang., segment control. MSDOS \$325

PS MAKE by Unipress - Interactive or batch. Full MAKE. MSDOS \$129

"C" LANGUAGE

	OUR PRICE
MSDOS C86-8087, reliable	call
Instant C - Inter., fast, full	445
Lattice C - the standard	call
Microsoft C 3.0 - new	279
Williams - debugger, fast	call
Wizard C - full, fast	450
CPM80 - EcoPlus C - faster, SLR	275
BDS C - solid value	125
MACINTOSH: Hippo II	375
Megamax - optimizer, full	275
Consular's MAC C, toolkit	395
Compare, evaluate, consider other Cs	

BASIC

	RUNS ON
BASCOM-86 - MicroSoft	8086 279
CB-86 - DRI	CPM86 419
Data Manager-full source	MSDOS 325
CADSAM-Full Btree, source	MSDOS 150
InfoREPORTER-multitile	PCDOS 115
Prof. Basic-Inter, debug	PCDOS 89
SCREEN SCULPTOR	PCDOS 115
TRUE BASIC - ANSI	PCDOS 125

Ask about ISAM, other addons for BASIC

SERVICE

ALL PRODUCTS - We carry 700 products for MSDOS, CP/M 86, CP/M 80. Macintosh and key products for other micros.

EDITORS Programming

	OUR RUNS ON PRICE
BRIEF - Intuitive, flexible	PCDOS 195
C Screen with source	86/80 75
Epsilon - like EMACS	PCDOS 195
FINAL WORD-for manuals	86/80 215
PMATE - powerful	8086 185
VEDIT - full, liked	86/80 119
XTC - multitasking	PCDOS 95

COBOL

	MSDOS	525
Dig. Res-decent	MAC	1850
MBP-Lev II, native, screen	MSDOS	885
Micro Focus Prof. - Full	PCDOS	call
Microsoft - Lev II, no royal	MSDOS	500
Ryan McFarland - portable	MSDOS	695

Ask about program generators.

LANGUAGE LIBRARIES

	MSDOS	95
GRAPHICS: Halo for Turbo Postal	PCDOS	125
GRAPHMATIC - 3D, FTN, PAS	PCDOS	220
MultiHALO - fast, full-all lang.	MSDOS	215
File MGNT: BTRieve-all lang.		86/80 369
Cindex + - source, no royal	ALL	369
CTree - source, no royal	8086	229
dBC ISAM by Lattice	MSDOS	465
dB VISTA - "Network" Structure	MSDOS	225
PHACT - up under UNIX, addons	MSDOS	129
OTHER: C Utilities by Essential	MSDOS	159
Greenleaf - 200 +	PCDOS	139
SOFT Horizons - Blocks I	PCDOS	125
SCREEN: CURSES by Lattice	PCDOS	139
MetaWINDOW - icons, clip	MSDOS	249
PANEL - many lang. term	PCDOS	415
ProScreen - windows, source	PCDOS	159
Turbo V - Greenleaf C. fast	MSDOS	175
Windows for C		

FORTRAN

	OUR RUNS ON PRICE
MS FORTRAN-86 - Impr.	MSDOS 239
DR Fortran-86 - full '77	8086 249
PolyFORTRAN-XREF, Xtract	PCDOS 165

OTHER PRODUCTS

	PCDOS	149
Advanced Trace 86 - Symbolic	8086	159
Assembler & Tools - DRI	PCDOS	395
Atron Debugger for Lattice	86/80	135
C Helper: DIFF, xref, more	PCDOS	125
CODESMITH-86 - debug	MAC	115
MacASM - full, fast, tools	8086	885
MBP Cobol-86 - fast	MSDOS	185
MicroProlog - improved	86/80	250
Micro: SubMATH - FORTRAN full	MSDOS	125
Microsoft MASM-86	PCDOS	265
Multitink - Multitasking	MSDOS	219
PC FORTH - well liked	MSDOS	345
Pinfish - Profile by routine	MSDOS	169
PL1-86 Debugger	8086	495
PL1-86	MSDOS	95
Polylibrarian - thorough	PCDOS	95
PolyMAKE	MSDOS	115
TRACE86 debugger ASM		
ZAP Communications-VT100, TEK 4010, full xfer	PCDOS	65

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and PDs. All formats available. UNIX is a trademark of Bell Labs.

Call for a catalog, literature, and solid value

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339

Visa MasterCard 8517

Listing Three

```

28: ;          status = dos( regsp );
29: ;          int      status ;          /* returned status register */
30: ;          REGS     *regp  ;          /* pointer to register struct */
31: ;
32: ; description: This function is a more useful version of the bdos()
33: ;              function provided with the lattice C compiler. It
34: ;              saves the existing machine state, replaces the
35: ;              contents of all registers except CS & SS from the
36: ;              structure pointed to by regp, does an int 21, loads
37: ;              the registers back into the structure, restores the
38: ;              machine state and returns.
39: ;
40: ; notes:      This function will only work with the SMALL model, since
41: ;              it assumes a 16 bit pointer. The stack frame is
42: ;              non-standard too (ie the BP holds the structure pointer
43: ;              arg rather than the frame pointer).
44: ;
45:          PSEG

46:          PUBLIC DOS
47: DOS      PROC      NEAR

48:          PUSH     BP                ; Save the old stack frame.
49:          MOV      BP,SP              ;
50:          MOV      BP,[BP+OFF]        ; BP = pointer to register structure.
51:
52:          PUSH     BX                ; Save the current machine state. No
53:          PUSH     CX                ; point in saving AX because it's
54:          PUSH     DX                ; going to be used for the return
55:          PUSH     SI                ; value.
56:          PUSH     DI
57:          PUSH     ES
58:          PUSH     DS

59:          MOV      AX,SS:[BP].AX_REG ; Set up a new machine state
60:          MOV      BX,SS:[BP].BX_REG ; using the contents of the
61:          MOV      CX,SS:[BP].CX_REG ; REGS structure. Don't modify
62:          MOV      DX,SS:[BP].DX_REG ; the SS or CS registers.
63:          MOV      SI,SS:[BP].SI_REG
64:          MOV      DI,SS:[BP].DI_REG
65:          MOV      ES,SS:[BP].ES_REG
66:          MOV      DS,SS:[BP].DS_REG

67:          PUSH     BP                ; Make the DOS call. Save the BP
68:          INT      21H              ; out of irrational paranoia.
69:          POP      BP

70:          MOV      SS:[BP].AX_REG,AX ; Now update the original structure
71:          MOV      SS:[BP].BX_REG,BX ; to reflect the return values from
72:          MOV      SS:[BP].CX_REG,CX ; the dos call.
73:          MOV      SS:[BP].DX_REG,DX
74:          MOV      SS:[BP].SI_REG,SI
75:          MOV      SS:[BP].DI_REG,DI
76:          MOV      SS:[BP].ES_REG,ES
77:          MOV      SS:[BP].CS_REG,CS
78:          MOV      SS:[BP].DS_REG,DS
79:          MOV      SS:[BP].SS_REG,SS

80:          POP      DS
81:          POP      ES
82:          POP      DI                ; Restore the previous machine state
83:          POP      SI
84:          POP      DX
85:          POP      CX
86:          POP      BX

```



```

87:          LAHF                      ; Return the flags:
88:          MOV      AL,AH             ;      Move flags to LSB
89:          XOR      AH,AH             ;      Clear high bytes

90:          POP      BP               ; Restore the previous stack frame's
91:          RET                      ; frame pointer and return.
92: DOS      ENDP

93: ; -----
94: ;
95: ; name:          gregs  --  Initialize a REGS structure to the current
96: ;                  register contents.
97: ;
98: ; synopsis:      #include <mydos.h>
99: ;
100: ;               gregs( regp )
101: ;               REGS      *regp      ;      /* pointer to register struct */
102: ;
103: ; description:    This function is used before before calling dos().
104: ;
105: ; notes:          This function will only work with the SMALL model.
106: ;

107:          PUBLIC  GREGS
108: GREGS     PROC    NEAR

109:          PUSH    BP                 ; Save the old stack frame.
110:          MOV     BP,SP               ;
111:          MOV     BP,[BP+OFF]         ; BP = pointer to register structure.

112:          MOV     SS:[BP].AX_REG,AX  ; Initialize the structure pointed
113:          MOV     SS:[BP].BX_REG,BX  ; to by bp.
114:          MOV     SS:[BP].CX_REG,CX
115:          MOV     SS:[BP].DX_REG,DX
116:          MOV     SS:[BP].SI_REG,SI
117:          MOV     SS:[BP].DI_REG,DI
118:          MOV     SS:[BP].ES_REG,ES
119:          MOV     SS:[BP].CS_REG,CS
120:          MOV     SS:[BP].DS_REG,DS
121:          MOV     SS:[BP].SS_REG,SS

122:          POP     BP                 ; Restore the previous stack
123:          RET                      ; frame and return.

124: GREGS     ENDP
125:          ENDPS
126:          END

```

End Listing Three

Listing Four

```

1: #include <stdio.h>

2: /*
3: *          PTEXT.C: Multi-column print utility
4: *
5: *          Copyright (c) 1985 Allen I. Holub,  all rights reserved.
6: *          This program may be reproduced for personal, non-profit use only
7: *
8: *          Bugs and Features:
9: *
10: *          1) Pr_line assumes that the output device supports backspace
11: *             ('\b' == ^H ).  This is only a problem when the source
12: *             has underlined text implemented as:
13: *                 texttexttext\r
14: *             We can't just output a '\r' in this case because we may
15: *             not be in the leftmost column. A \r should get us to the
16: *             left edge of the current column.
17: *

```

(Continued on next page)

Listing Four

```

18: *      2) When printing multi-column stuff. If a line is truncated it
19: *      will run into the column to its right. That is, there is no
20: *      seperator between columns other than the whitespace needed
21: *      to pad the column out to a particular width. If no padding
22: *      is necessary (ie. the lines have been truncated), then the
23: *      columns will run together.
24: *
25: *      3) When an ESC is found, the escape and the next three characters
26: *      take up no space in the output. This lets us print out the
27: *      various SGR commands without messing up the column width.
28: *
29: *      4) Strange things happen to an IBM screen when the leftmost
30: *      character on a line is printed with underline. To compensate
31: *      for this, a single blank is printed as the left-most
32: *      character on every line if IBM is #defined.
33: */

34: #define ESC      0x1b
35: #define IBM      1

36: /*-----*/

37: ptext(linec, linev, outfile, numcols, colwidth, numRows)
38: int      linec, numcols, colwidth;
39: char     **linev;
40: FILE     *outfile;
41: int      numRows;
42: {
43:     /*      Print out the array of strings "linev" which consists of
44:     *      linec entrys. Output is sent to "outfile" formatted as
45:     *      follows:
46:     *
47:     *      "numcols" : number of columns
48:     *      "colwidth" : width of a column in characters. Any text
49:     *                  longer than colwidth is truncated off.
50:     *      "numrows" : columns are numRows long. The left-most
51:     *                  column is printed in its entirety
52:     *                  first then the next column in its
53:     *                  entirety, and so on.
54:     */

55:     register int      j ;
56:     register char     **lineend, **line, **nextline ;

57:     lineend = &linev[linec-1];      /* Last linev entry to print */

58:     for( j = numRows ; --j >= 0 ; )
59:     {
60: #ifdef IBM
61:         putc( ' ', outfile );
62: #endif
63:         for( line = linev++; line <= lineend; line = nextline)
64:         {
65:             nextline = line + numcols;

66:             /*      Print the line. Don't pad the rightmost
67:             *      column.
68:             */
69:             pr_line( *line, outfile, colwidth,
70:                     nextline <= lineend);
71:         }

72:         fputs("\n" , outfile);
73:     }
74: }

```

(Continued on page 40)

BRIEF™

"BRIEF Is The Best Editor I Have Ever Used. I Switched. I Use BRIEF Regularly."

Steve McMahon*

Choosing the right program editor is an extremely important decision. After all, more time is spent with your program editor than with any other category of software. A good program editor helps to stimulate creativity and produce programs of superior design. A poor program editor, on the other hand, can be worse than none at all. It can slow you down, get in your way, and make even the simplest tasks difficult.

BRIEF - PROGRAM EDITING YOUR WAY

Every programmer has an individual style that makes their work unique. Most program editors compromise that style by forcing the programmer to conform to their methods. Not so with BRIEF.

BRIEF's most powerful feature is its ability to conform to your way of programming. BRIEF can be easily tailored/customized to your individual preferences. For example, keyboard reassignment allows the keyboard to be used in whatever way you prefer. Keystroke macros allow long sequences of editor directives to be repeated by pressing a single key. Plus you can simultaneously work with numerous program and data files, and configure many windows to control BRIEF's visual presentation.

75% OF THE EXPERTS WHO HAVE TRIED BRIEF SWITCHED!

(Call For Details)

```
make.c
int handle = 0;
main (argc, argv)
int argc;
#include "fsa.h"
typedef struct
{
    short action,
    state;
} Fsa;
#ifdef FSA_MAIN
Fsa fsa[10] = {0}; /* Alphanum Co
/* State 0. */ 0, 2, 10
/* State 1. */ 10, 0, 10
/* State 2. */ 0, 2, 1
/* State 3. */ 0, 5, 11
/* State 4. */ 0, 4, 0
}

makefile.h
/**
** makefile.h:
** This is the definitions fil
** Hopefully, it won't be unreasonab
** that have been written.
**/
typedef struct cmd_struct
{
    char *cmd_text;
    struct cmd_struct *next_cmd;
} *Cmd_Ptr, Cmd;
```

Mismatched open parenthesis.

Line: 11 Col: 17 2:17 PM

BRIEF'S ONLY LIMITATIONS ARE THE ONES YOU SET

BRIEF's capabilities are limited only by your imagination. Once you've used BRIEF, you will NEVER want another editor.

TRY BRIEF RISK FREE FOR 30 DAYS!

Order BRIEF (\$195) and try it for 30 days with a full money-back guarantee. CALL 800-821-2492

"A BONA FIDE UNDO" Steve McMahon - BYTE REVIEW "BUILD YOUR DREAM EDITOR" MARCH 1985

Here are Steve McMahon's exact words: "...BRIEF implements a true undo facility, by default allowing command-by-command recovery from the last 30 commands...The number of commands you want to be able to undo can be changed." (up to 300) "Only with BRIEF, though, was it possible to undo a macro that produced 4000 words of text with a single keystroke."

EVERY FEATURE YOU WANT

BRIEF supports practically every feature you've ever seen, and some you probably haven't thought of yet.

- Edit Multiple Large Files
- Windows (Tiled & Pop-up)
- Unlimited File Size
- Full UNDO (N Times)
- True Automatic Indent for C
- Exit to DOS Inside BRIEF
- Compile programs inside BRIEF
- Uses All Available Memory
- Intuitive Commands
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning
- Reconfigurable Keyboard
- Online Help
- Search for "Regular Expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- And ... a Complete, Compiled, Programmable and Readable Macro Language.

A TYPICAL BRIEF SCREEN

Notice there are three windows on the screen simultaneously and each one is viewing a different file. The mainline of a C program is visible in the uppermost window; the programmer has run a syntax checking macro which found a mismatched open parenthesis in the arguments to the mainline. The other two windows show header files containing information crucial to the design of the program. BRIEF can have an unlimited number of windows and files accessed simultaneously.

AVAILABILITY: IBM-PC and Compatibles, AT, & TANDY 2000

Solution Systems™

335-D Washington St., Norwell, MA 02061 (617) 659-1571

*Steve McMahon's quote courtesy Suntime Publishing Systems. BYTE review by Mr. McMahon may be found in BYTE Magazine March 1985.

Listing Four

```
75: /*-----*/
76: static pr_line( str, stream, width, padded )
77: register char *str;
78: FILE *stream;
79: int width, padded;
80: {
81:     /* Print out "str" into "stream" padding it to "width"
82:     * columns wide. Non-printing characters and 3 printing characters
83:     * immediatly following an ESC are not counted as having printed.
84:     * '\n' characters are treated as line teminators but are not
85:     * printed. If "padded" is 0 then no padding is done, though the
86:     * line will still be truncated if it's too long.
87:     */
88:     int col = 0 ;
89:     while( col < width && *str )
90:     {
91:         if ( *str == '\n' )
92:             break;
93:         else if( *str == '\r' )
94:         {
95:             /* Back up to the left edge of the current column
96:             */
97:             while( col > 0 )
98:             {
99:                 --col;
100:                putc( '\b' , stream);
101:            }
102:            str++;
103:        }
104:        else if( *str == '\t' ) /* expand tabs */
105:        {
106:            str++ ;
107:            col++ ;
108:            putc( ' ' , stream );
109:            while( (col % 8) && col < width)
110:            {
111:                putc( ' ' , stream );
112:                col++;
113:            }
114:        }
115:        else
116:        {
117:            if( *str == ESC )
118:            {
119:                putc( *str++ , stream );
120:                if( !*str )
121:                    break;
122:                putc( *str++ , stream );
123:                if( !*str )
124:                    break;
125:                putc( *str++ , stream );
126:                if( !*str )
127:                    break;
128:            }
129:            else if ( *str == '\b' )
130:                --col ;
131:        }
```



```

132:                else if( *str >= ' ' )
133:                    ++col ;

134:                putc( *str++ , stream );
135:            }
136:        }

137:        if (padded)
138:            while( col++ < width )
139:                putc( ' ' , stream );
140:    }

```

End Listing Four

Listing Five

```

1:  /* Example of how to get and restore the MSDOS Disk Transfer address
2:  *  using the dos() and gregs() functions defined in listing 3
3:  */

4:  #include <mydos.h>

5:  #define SETDTA  0x1a
6:  #define GETDTA  0x2f

7:  main()
8:  {
9:      static FILE_INFO info;
10:     static REGS      oregs, nregs;

11:     gregs( &oregs );                /* Get the old DTA. It will be put */
12:     oregs.h.ah = GETDTA;            /* into ES:BX of oregs.          */
13:     dos( &oregs );

14:     gregs( &nregs );
15:     nregs.x.dx = (word) &info;      /* Change the Disk Transfer Addr  */
16:     nregs.h.ah = SETDTA ;           /* to point at info structure     */
17:     dos( &nregs );

18:     /*      . . .      */

19:     oregs.x.ds = oregs.x.es;        /* Now put the DTA back. You have */
20:     oregs.x.dx = oregs.x.bx;        /* to copy ES:BX into DS:DX first. */
21:     nregs.h.ah = SETDTA ;
22:     dos( &oregs );
23: }

```

End Listings

Build a Custom PC or Clone

by Jim Kronman

I have owned an 8-bit S-100 microcomputer since 1978. Old habits are hard to kick, so I set about assembling a PC-compatible computer the same way I built my S-100 system. I individually selected each component of the system to suit my specific needs. I now have a PC XT work-alike that is totally "IBM compatible," costs substantially less than the closest off-the-shelf equivalent, and fits my requirements exactly.

My approach may not be for everyone. I am pleased with my completed project, but you should consider the pros and cons before taking the plunge yourself. Before going into the advantages of rolling your own PC, I'll point out some of the drawbacks you should consider.

with microcomputer hardware or are timid when handling equipment costing, in the aggregate, up to several thousand dollars. When you put Manufacturer A's board and Manufacturer B's board into your computer, you alone are responsible for determining that the boards are compatible both with each other and with the PC itself. The vendors and manufacturers usually will try to help if you have a problem, but they may never have encountered exactly the configuration you have assembled and therefore may be unable to determine the cause of your problem.

Now that you have received fair warning of some of the potential pitfalls, I'll shamelessly promote the advantages of building your own PC.

***Careful shopping, gentle persuasion with vise grips
and a lot of patience yield an XT work-alike
at a PC price.***

You will need patience. If you must have a computer tomorrow, go buy something off the shelf of your local computer store. I ordered pieces from five separate mailorder suppliers (including IBM), and it was over five weeks from the time I placed the orders by telephone until I first booted the system. At that point, I was still missing a vital element, but more on that subject later.

Do not attempt to assemble your own PC if you are not experienced

You can save a bundle of money when you compare your homegrown computer to one with the same functionality in a store: for the same amount of money, you can get a lot more capability. With my project now complete, I realize, too, that there are other benefits I had not anticipated when I began.

An obvious reason to undertake this adventure is to assemble a system exactly suited to your needs. One of my prime requirements was to have a Selectric style keyboard, which IBM does not offer with the PC. I did not want to buy the standard IBM PC because I would end up throwing away a perfectly good (if poorly laid out) key-

*Jim Kronman, 6085 Venice Blvd.,
No. 16, Los Angeles, CA 90034 (213)
558-3281*

board and buying a replacement. (As far as I know, IBM will not sell the PC System Unit without a keyboard.)

I purchased the Keytronic KB5151 deluxe keyboard with a Selectric style layout and separate numeric and arrow keypads; it also has the function keys across the top of the keyboard so they will align with the function key legends in row 25 of the display in many application programs. My one criticism of the Keytronic design was that the "touch" was too soft; sometimes the slightest tap would cause a character to appear on the screen. This problem was easy to remedy. I carefully took each keytop off of the keyboard, stretched the spring under the keytop to about 0.8 inches, and then replaced the keytop. Do one key at a time and you should have no problems with this slight modification; it's well worth the time!

I had decided that I wanted the capability of a hard disk and that I wanted it internal to the computer. I also did not like the crowded arrangement in the eight-slot PC XT or the XT clones. These considerations and others led me to opt for a custom assembly with a five-slot PC motherboard. I'm sure you can think of plenty of your own reasons to customize, too.

While my project was in progress, I had occasion to help a friend with an IBM PC that had a failed power supply. What I have come to call the "\$290 Fuse Syndrome" is a strong argument to buy as few components as possible from IBM. Big Blue's warranty policy (which I have seen reviewers praise in several magazine articles) is to replace defective components on the spot and without question. That's great while you are within the warranty period and the tab is on Blue, but unfortunately the policy continues and is *mandatory* after the warranty expires and you are paying the bill. If you try to obtain a schematic or other service information for an IBM PC component that is designated a FRU (Field Replaceable Unit), you will encounter as firm a stonewall from IBM as you will ever want to see. Even if you figure out who made the component for IBM, you can't get any help because IBM

has its vendors contractually bound to stony silence as well.

The \$290 Fuse Syndrome is so named because inside of each IBM power supply is a standard slow-blow radio-type fuse—in a fuse holder! If this fuse blows (or even simply fails), IBM's policy is that the supply is terminally (if you'll pardon the expression) ill and must be replaced. A PC XT (130 watts) power supply costs \$290 when ordered from IBM; as I write this article, the going rate for a clone replacement seems to be around

\$170. By purchasing your power supply, plug-in boards, disk drives, and so on from other sources, you can obtain servicing information in most cases. If the availability of servicing information is vital to you, check with each manufacturer before you buy.

My original plan was to purchase a PC motherboard, the case, and a 130 watt power supply from IBM. This would have given the machine the appearance of being "true Blue." IBM was one step ahead of me: I was unable to obtain the entire case from

Advanced Screen Management made easy

Now a professional software tool from
Creative Solutions.

WINDOWS FOR C™

More than a window display system,
WINDOWS FOR C is a video tool kit for all
screen management tasks.

- Pop-up menus and help files
- Unlimited files and windows
- Instant screen changes
- Complete color control
- Horizontal and vertical scrolling
- Word wrap
- Highlighting
- Auto memory management
- Plus a library of over 50
building block subroutines

**Designed for portability.
Easy to learn, easy to use.**

Once you've tried WINDOWS FOR C,
you'll wonder how you ever managed without it.

Full support for IBM PC/XT/AT and compatibles, plus interfaces for non-IBM computers;
Lattice C, C1-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS),

NEW Ver. 3.1

Enhanced portability.
Topview compatible.

WINDOWS FOR C \$195
(specify compiler & version)

Demo disk and manual \$ 30
(applies toward purchase)

**Full source available.
No royalties.**



Creative Solutions

21 Elm Ave., Box T7,
Richford, VT 05476

802-848-7738

Master Card & Visa Accepted
Shipping \$2.50
VT residents add 4% tax.

Circle no. 27 on reader service card.

IBM, even though all the pieces are listed in the IBM PC Service Manual's illustrated parts breakdown and the Greencastle Parts Depot accepted my order. I did buy the power supply, which has given me no problems to date (although the first supply sent to me was dead on arrival), but I regret not saving the \$100 by buying a supply from a third party.

My inability to obtain the case from IBM was frustrating; after the IBM Parts Center accepted my order for all of the pieces, it simply failed to deliver all of them. I never received notification that the entire order would not be filled. As the weeks dragged by, I assembled all of the components of the by now functional computer on a 24 × 27-inch piece of particle board. When I finally found out what was happening with my order from IBM, the Parts Depot at least let me return for credit the pieces of the case I had obtained and could not use.

I found a clone chassis made in Taiwan for about half the price of the IBM parts, and after a small amount of persuasion with vise grip pliers to straighten out a bracket, everything fit fine. The case is identical in appearance to the IBM from the front, except for the absence of the famous logo, and the back panel has a variety of cutouts adaptable to various hardware configurations.

One piece of the system that had to be genuine IBM was the motherboard, which contains the IBM ROM BIOS, thus assuring total PC compatibility. Although some of the clone boards might be close to 100% compatible, the difference in price was not worth the risk. If you have reliable information about a specific non-IBM motherboard that meets your needs, then go for it!

One other specific piece of hardware made the choice of the PC (rather than XT) motherboard attractive: Maynard Electronics' "Sandstar Series" WS-2 disk controller board supports up to four floppy disk drives and two 10 Mb hard disks and takes up only a single slot. An added advantage of this configuration is that the Maynard hard disk software driver is in a PROM that is installed on the main

circuit board, occupying address space in the F0000h segment, while the IBM XT controller is a separate board and has its driver in the middle of the C0000h segment. Thus, by not using the XT controller or any other board with ROM code in the C0000h segment, you make available 64K of memory space along with the D0000h and E0000h segments, which allows you to install a 192K RAM disk without using any of the 640K available to DOS. Use caution in dealing with the C0000h through E0000h segments; according to the IBM Technical Reference Manual (page 2-11), plug-in boards can use the area from C2000h to F4000h for software drivers that the ROM BIOS will recognize at power-on.

When you do not buy the System Unit from IBM, you do not receive the IBM PC Guide to Operations manual, the BASIC manual, or the IBM Diagnostic Disk. You may purchase these items from IBM if you need them. If you purchase the IBM Technical Reference Manual, you will not need the Guide to Operations. (If you need this Guide to Operations, you probably shouldn't be assembling your own computer!) I highly recommend that you purchase the Technical Reference Manual; it is the definitive source of information on the ROM BIOS function calls, containing the source code for the entire ROM BIOS.

After I completed my project (and after my friend purchased a replacement power supply), I discovered that Howard W. Sams publishes a "Computerfact" package for the IBM PC. It costs \$39.95 and includes schematics, circuit board photos with component identifications and trouble-shooting instructions. The package covers the PC motherboard, power supply, keyboard, display adapters, floppy disk controller, and a few other optional cards.

You must also purchase an operating system separately. The various choices are PCDOS 2.1 or 3.0 from IBM and Digital Research's CP/M-86, Concurrent CP/M v3.1, or Concurrent DOS v3.2. By far the most software is available for PCDOS. Because v3.0 seems to offer the single

user little advantage over v2.1, I selected PCDOS 2.1. The choice is yours. You can use more than one operating system (but not simultaneously).

That is about all there is to tell about my experiences. Not only would I do it again this way if I had to (less the mistakes, of course), I would not consider doing it any other way! What follows is a summary of my advice and some useful tabulations of information, components, and sources.

If you intend to assemble your own PC:

- (1) Be confident of your skills and knowledge in microcomputers.
- (2) Be prepared to wait; this isn't a project for the impatient.
- (3) Select your components carefully; make sure each item is compatible with the others.
- (4) Understand each vendor's return policy before you order, just in case.
- (5) Select your software with as much or more care as your hardware.
- (6) If you have a specific software application in mind, make sure the hardware you select is compatible.

The components you will need (as a minimum) are:

- (1) Motherboard, with CPU, ROM BIOS, memory, keyboard interface, and support circuits
- (2) Power supply, 62 to 130 watts (130 watts recommended)
- (3) Display adapter card
- (4) Monitor compatible with display adapter
- (5) Disk controller card (possibly a second card for hard disk)
- (6) Case
- (7) Keyboard
- (8) Floppy disk drive, 5¼-inch DSDD, 48 tpi, PC-compatible

Optional components include (but are not limited to):

- (1) Multifunction card (additional memory, battery run clock, serial and parallel ports, game port, etc.)
- (2) MODEM card
- (3) Math coprocessor chip (8087)
- (4) Coprocessor board (Z80 CP/M-80, Apple, 10286, etc.)

The following items are available from IBM:

- (1) PC motherboard with 64K memory installed, P/N 8654213, \$640.00
- (2) PC XT motherboard with 128K memory installed, P/N 8529254, \$750.00
- (3) Speaker with mounting bracket, P/N 8529143, \$8.15
- (4) Guide To Operations manual
- (5) Technical Reference Manual, P/N 1502234
- (6) BASIC reference manual, P/N 6025010
- (7) PC DOS 2.1, P/N 6024120
- (8) PC Diagnostic Disk, P/N 1502212

All of the above except for the motherboards and speaker should be available through the IBM Product Centers or other retail outlets (Sears Business Centers are usually a good place to check). The replacement parts (motherboard, etc.) are available from the IBM Parts Depot in Greencastle, IN. It is best to deal by telephone: (317) 658-2022. The address is P.O. Box 505, Greencastle, IN 46135. The hardware prices may change; they are included here for reference only.

For the other pieces that you will need for your system, look through the ads in this magazine, *The Computer Shopper*, and *Byte*. Many firms sell PC clone cases, power supplies, keyboards, and so on. I have even found bargains at surplus companies; a firm in the Los Angeles area recently had surplus Keytronics keyboards manufactured with special legends on the keys for just \$35!

Happy hunting and happy cloning. I hope this article encourages fellow computer hobbyists to explore the PC bus in the same spirit as the S-100 bus. I have found the experience to be well worth the effort.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 192.

How to go from UNIX to DOS without compromising your standards.

It's easy. Just get an industry standard file access method that works on both.

C-ISAM™ from RDS.

It's been the UNIX™ standard for years (used in more UNIX languages and programs than any other access method), and it's fast becoming the standard for DOS. Why?

Because of the way it works. Its B+ Tree indexing structure offers unlimited indexes. There's also automatic or manual record locking and optional transaction audit trails. Plus index compression to save disk space and cut access times.

How can we be so sure C-ISAM works so well?

We use it ourselves. It's a part of INFORMIX®, INFORMIX-SQL and File-it!™, our best selling database management programs.

For an information packet, call (415) 424-1300. Or write RDS, 2471 East Bayshore Road, Palo Alto, CA 94303.

You'll see why anything less than C-ISAM is just a compromise.



RELATIONAL DATABASE SYSTEMS, INC.

© 1985, Relational Database Systems, Inc. UNIX is a trademark of AT&T Bell Laboratories. INFORMIX is a registered trademark and RDS, C-ISAM and File-It! are trademarks of Relational Database Systems, Inc.

The Ultimate Parallel Print Spooler

by Don Rindsberg

Don Rindsberg has come up with a printer spooler design incorporating low-cost 6665 (64K) dynamic RAM chips. The result—a 60K print spooler that you can build for about a hundred dollars.

If you are tired of waiting while your parallel printer does its thing, here is a project that will allow you to print and compute at the same time. It is a print spooler that provides a whopping 60K of buffering. The spooler, designed to stand alone, will interface almost any parallel printer port.

The project incorporates “software refresh” of the dynamic RAM memory—a concept little publicized, to my knowledge, except in Motorola’s application literature. By using software refresh of RAM, you can eliminate a number of integrated circuits from the design of the spooler. The refresh cycle

I built this spooler to interface my Apple II computer to an Epson parallel printer. However, it should be easily adaptable to virtually any computer with a parallel printer port and any parallel printer. The 60K of buffering holds at least 12 pages of solid, single-spaced text. In the graphics mode, an entire screen is dumped to the spooler in a couple of seconds.

The Hardware

The 6809 microprocessor, which is now available for about \$12, is ideal for our software-refreshed memory system: its internal clock can provide the time-critical signals to the dynamic memory without the use of additional clocks or delay lines. The 6809 (not the 6809E) has its own on-chip clock generator and requires only a crystal in the 3.5 to 4.0 MHz range; a low-cost TV color-burst crystal is a

This design uses software refresh of dynamic ROM to create a 60K spooler for about \$100.

ties up the microprocessor for less than 10 percent of its time, a penalty easily tolerated when you are controlling a printer, which operates at a snail-like speed from the microprocessor’s point of view. You implement software refresh by carefully writing the software so that the 128 consecutive row addresses are read at least every 2 ms: 128 rows in a dynamic RAM must be refreshed, and the maximum allowable time between refreshes is 2 ms.

Don Rindsberg, 5958 S. Shenandoah Rd., Mobile, AL 36608.

simple and inexpensive choice.

The 6809 produces two clock signals, E (enable) and Q (quadrature), which are both 50 percent duty cycle square waves at a frequency exactly one-fourth that of the crystal. As shown on the timing diagram (Figure 1, page 47), the Q signal leads the E signal by one-fourth of a cycle—the E signal, by the way, is the same as the phase two signal on the 6800 and 6502 microprocessors. We use the rising edge of Q to trigger RAS (row address strobe) on the dynamic RAM. The rising edge of E causes the switch from row addresses to column

addresses, while the falling edge of Q triggers CAS (column address strobe) to the RAM chips. RAS is exerted every cycle to force a refresh, while CAS is exerted only if the address is not an "F" address (\$F000–FFFF).

Figure 2 (page 48) shows that we decode address F in the 74LS20, which produces a low when A12–A15 are all high. The 2716 ROM is at \$F800–FFFF, and the PIA (parallel interface adapter) is at \$F400–F403; address 11 provides the distinction between the ROM and the PIA. We buffered A11–A15 with a 74LS367 because the 6809 has limited drive capabilities. Figure 3 (page 48) shows how the switching between the row addresses A0–A7 and column addresses A8–A15 was done in a pair of 74LS157 chips. The 33 ohm resistors prevent ringing on the RAM's address lines.

The 6821 PIA in Figure 4 (page 50) was designed for applications such as this, having exactly the number of functions we need. It automatically provides the handshaking signals used by parallel printers. When the A side is used as a parallel input, STR from the host computer sets a flag. When the spooler reads the data, the PIA automatically provides a one-cycle negative-going signal, which is used for ACK. On the B side of the PIA, a similar STR signal automatically is generated on a write to port B, and the printer's ACK signal sets a flag to signal that the printer is ready for another character.

Figure 5 (page 50) shows the completed spooler. A 74LS74, used in conjunction with the pause switch, allows manual shutdown of the printout to change paper and so on. This chip changes state at the falling edge of E when you change the switch position. At this time, the opposite half of the ROM is bank-switched into memory, which contains a holding routine. The two halves of the ROM are selected by its A10 line. The power supply (Figure 6, page 51) provides regulated 5 volts at somewhat less than 500 ma.

The Software

The software has two unusual requirements:

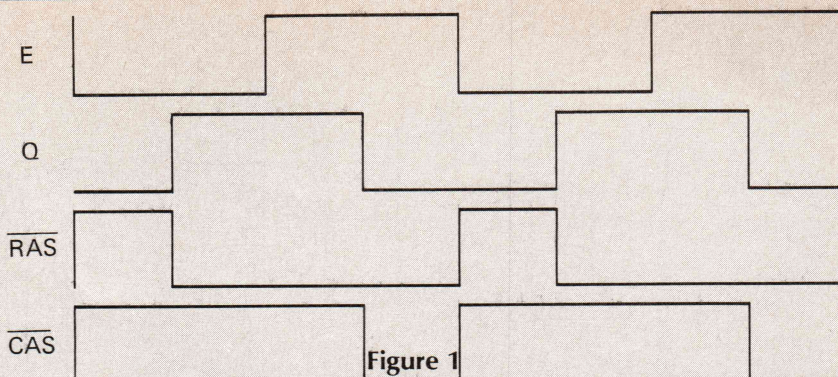
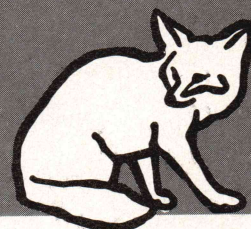


Figure 1

The timing diagram shows how the signals E and Q from the 6809 gate the column (CAS) and row (RAS) logic in the 6665 dynamic memory chips. This eliminates the need for delay circuits.

dBASE II® Source compatible

FoxBASE



Evolution.

Now FoxBASE, the dBASE II source-compatible interpreter/compiler, is even better than before. Automatic 8087 co-processor support allows you to perform numeric computations with lightning speed. Fourteen-digit precision gives you 40% greater accuracy than dBASE II. The fact that FoxBASE is not copy protected means you can easily load it onto your hard disk. What's more, FoxBASE comes complete with a NO-RISK demo plan.

Of course, FoxBASE still offers all of the features that dBASE II does . . . PLUS

- Runs 3 to 20 times faster • Permits up to 48 fields/record...50% more than dBASE II
- Supports full type-ahead • Compiles pro-

gram sources into compact object code • Has twice as many variables • Comes with a sophisticated online manual and HELP facility.

FoxBASE is currently available on a wide range of machines: IBM-PC, IBM-PC/XT/AT, COMPAQ & IBM compatibles, TI Professional, DG Desktop, and DG MV-Series to name just a few. And it will soon be available on the Molecular and NCR Tower computers as well. Call or write today for more information.

MS-DOS: Development Pkg. **\$395**

Runtime Pkg. **\$695**

AOS/VS: Development Pkg. **\$995**

Runtime Pkg. **\$1995**

UNIX and XENIX: (To Be Announced)

Developed by

DACOR

COMPUTER SYSTEMS

dBASE II is a trademark of Ashton-Tate.
UNIX is a trademark of AT & T.
FoxBASE is a trademark of Fox Software Inc.

FoxBASE™
from FOX SOFTWARE INC.



13330 Bishop Road, P.O. Box 269, Bowling Green, OH 43402 / 419-354-3981 / TWX 810-499-2989

Circle no. 40 on reader service card.

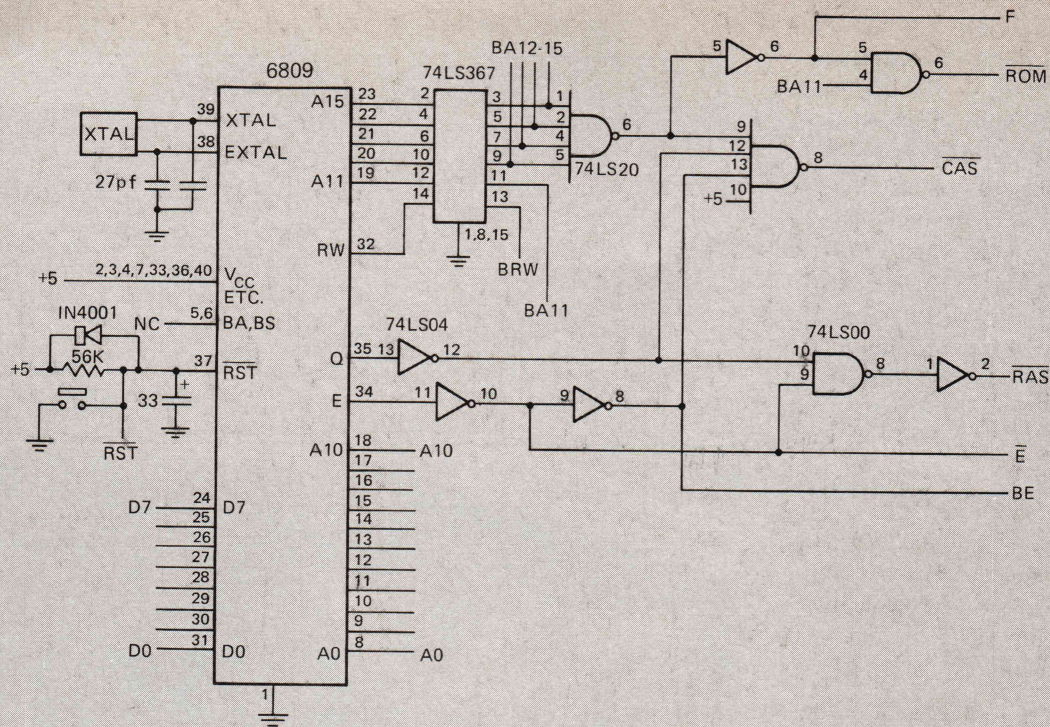


Figure 2.
The wiring diagram shows the 6809 and the address decoding. A 3-4 MHz crystal is required (frequency is not critical).

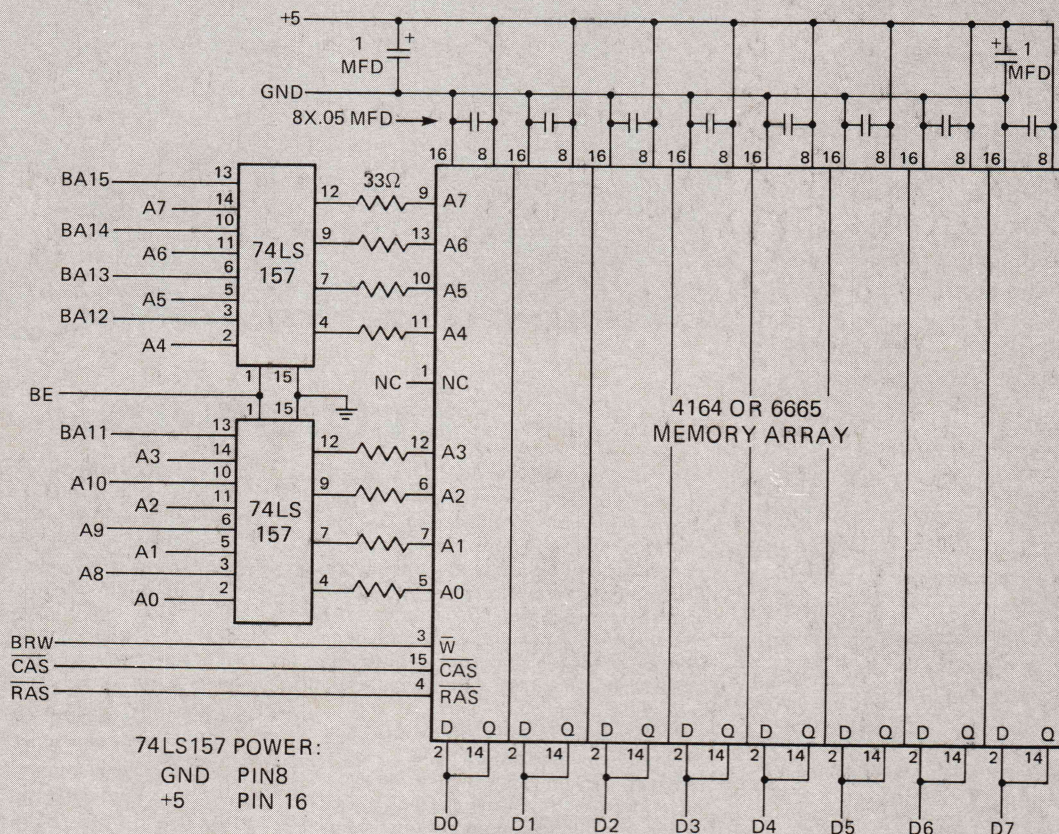


Figure 3.
The eight dynamic RAM chips are wired in parallel. Make identical connections to each chip with the exception of the data lines; a different data line connects to each chip as shown. Be sure to install a decoupling capacitor near each 6665.

PROGRAMMER DEVELOPMENT TOOLS

NEW LISTINGS:

List Ours

Brief By Solution Systems	195	Call
Megamax C compiler for Macintosh	295	239
PC Lint by Gimpel Software	100	89
Prolog-86 by Solution Systems	125	Call
Scientific Subroutine Lib for C by Peerless	175	159

C-terp Complete C Interpreter

Full K&R C interpreter/semi-compiler which can access functions and externals compiled on various C compilers. Comes with a powerful, integrated screen editor providing a complete professional C programming environment.

List Price \$300 Our Price **\$269**

C LANGUAGE:

Computer Innovations C-86 Compiler	395	299
DeSmet C Compiler with Debugger	159	145
Lattice C Compiler from Lattice	500	339
Lattice C from Lifeboat	500	275
Mark Williams C Compiler w/Source Debugger	495	429
Run/C Interpreter by Age of Reason	150	99
Safe C Standalone Interpreter by Catalytix	Call	Call
Wizard C Compiler by Wizard Systems	450	399
Xenix Development System by SCO	1350	1099

Microsoft C Compiler version 3.0

This entirely new version of Microsoft's C compiler features fast execution and compact code generation, small, medium and large memory models, Xenix compatibility, a linker and a librarian.

List Price \$395 Our Price **\$339**

OTHER LANGUAGES:

8088 Assembler w/Z-80 Translator by 2500 AD	100	89
APL+PLUS/PC by STSC	595	469
BetterBASIC by Summit Software	200	169
Golden Common LISP by Gold Hill	495	Call
Janus/ADA by R&R Software	900	699
MASM-86 ver 3.0 w/utilities by Microsoft	150	119
Modula-2/86 by Logitech	495	439
Professional BASIC by Morgan Computing	99	89
RM/Fortran by Ryan-McFarland	595	439

C UTILITIES:

C Power Paks From Software Horizons	Call	Call
C-Sprite Symbolic Debugger for Lattice C	175	159
c-tree by FairCom	395	359
C Utility Library by Essential Software	Call	Call

C UTILITIES:

DBC dBase/C Interface by Lattice	250	219
DBC with source code	500	459
ESP for C by Bellesoft	Call	Call
GraphicC by Scientific Endeavors	250	209
Greenleaf C Functions Library ver 3.0	185	139
Greenleaf Comm Library	185	139
Multi-Halo Graphics by Media Cybernetics	250	199
PANEL Screen Designer ver. 6.0 by Roundhill	295	234
Pasm86 Macro Assembler by Phoenix	295	259
Pfinish Performance Analyzer by Phoenix	395	339
Pmaker Program Development Manager by Phoenix	195	179
Pre-C Lint Utility by Phoenix	395	339
Safe C Dynamic Profiler by Catalytix	150	Call
Safe C Runtime Analyzer by Catalytix	400	Call
Windows For C by Creative Solutions	195	139

TURBO PASCAL:

Screen Sculptor by Software Bottling	125	109
Turbo ASYNCH by Blaise Computing	100	89
Turbo GRAPHICS TOOLBOX by Borland Int'l	55	49
Turbo PASCAL ver 3.0 by Borland Int'l	70	49
Turbo PASCAL w/8087 or BCD	110	89
Turbo PASCAL w/8087 & BCD	125	99
Turbo POWER TOOLS by Blaise Computing	100	89
Turbo TOOLBOX by Borland Int'l	55	49
Turbo TUTOR by Borland Int'l	35	29
XTC Text Editor by Wendin	99	89

OTHER PRODUCTS:

Advanced Trace-86 by Morgan Computing	175	159
APL2C by Lauer Software	150	139
Blaise Tools for C & Pascal	Call	Call
Btrieve by SoftCraft	250	199
Codesmith-86 Debugger by Visual Age	145	129
Epsilon Emacs-like Text Editor by Lugu	195	179
FORTRAN Libraries by Alpha Computer Service	Call	Call
Pfix-86 Plus by Phoenix	395	299
Plink-86 Overlay Linker by Phoenix	395	299
Pmate Macro Text Editor by Phoenix	225	159
Polytron Products	Call	Call
Profiler by DWB Associates	125	89
Xtrieve by SoftCraft	195	169

Periscope Symbolic Debugger by Data Base Decisions

Write-protect memory board and breakout switch allows instant recovery from runaway code. Provides on-line help, windowing, extensive breakpoints, dual monitor support and more.

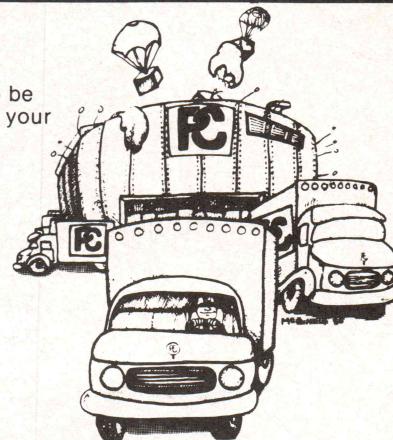
List Price \$295 Our Price **\$269**

ADVANTAGE #2

At Programmer's Connection most popular products are always in stock ready to be shipped. Most orders are on their way to you the same day they are placed. We know your time is important, that's why we tell you exactly when you can expect to receive your package. Maintaining an adequate inventory is part of our philosophy of fast, efficient service. Call us — you'll discover that Programmer's Connection delivers products and service without delay.

Discover the advantages of buying from Programmer's Connection:

1. We offer the latest version of a product.
2. Most popular products are in stock ready to be shipped.
3. Receive same manufacturer's support as if buying direct.
4. Experienced professional programmers are on staff.
5. Choose from a large selection of the best software products available.
6. Knowledgeable and courteous sales staff.
7. Significant discounts off of retail prices.
8. No extra charge on prepaid orders, including major credit cards.
9. Reasonable charges for shipping and handling.
10. Toll free services from Canada and the U.S.



"Programmers Serving Programmers"



Programmer's Connection

136 Sunnyside Street Hartville, Ohio 44632 (216) 877-3781 (In Ohio)

U.S.; **1-800-336-1166** Canada; **1-800-225-1166**

Call For Our Catalog



Account is charged when order is shipped
Prices are subject to change without notice

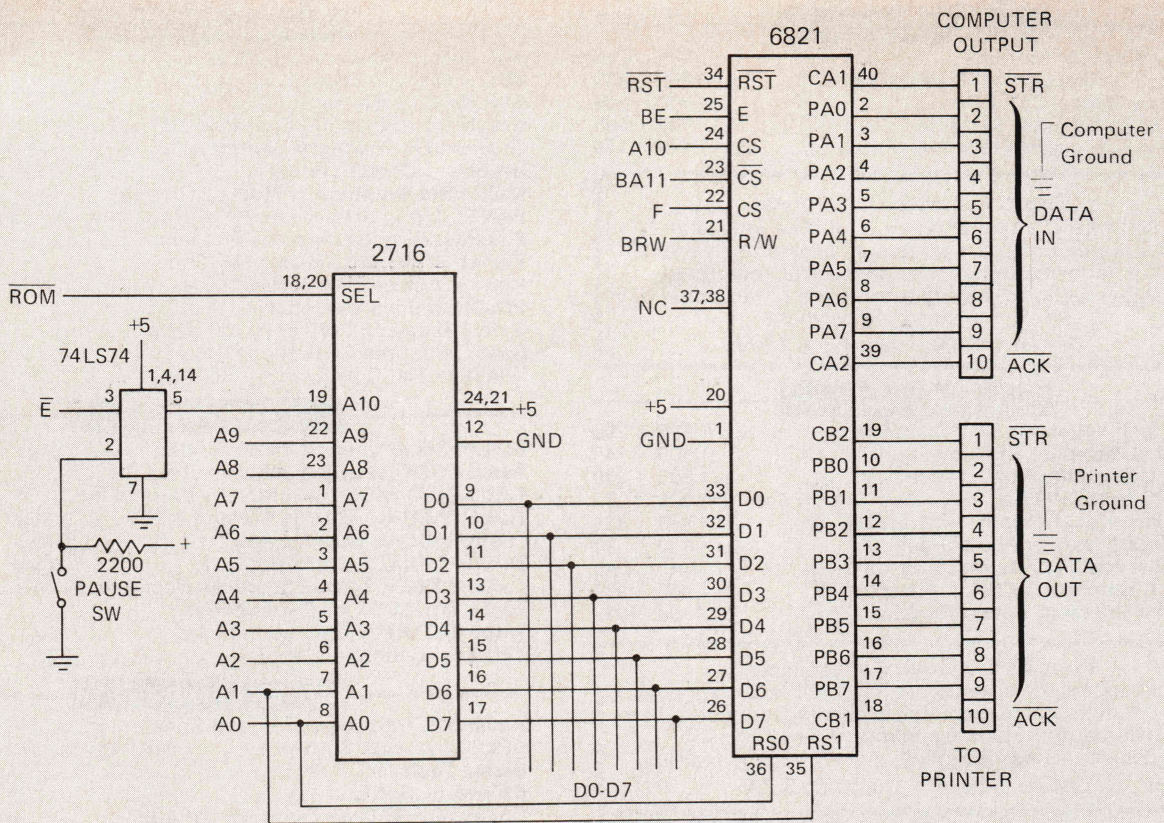


Figure 4.
Wiring of the 2716 EPROM and the 6821 parallel interface adapter.

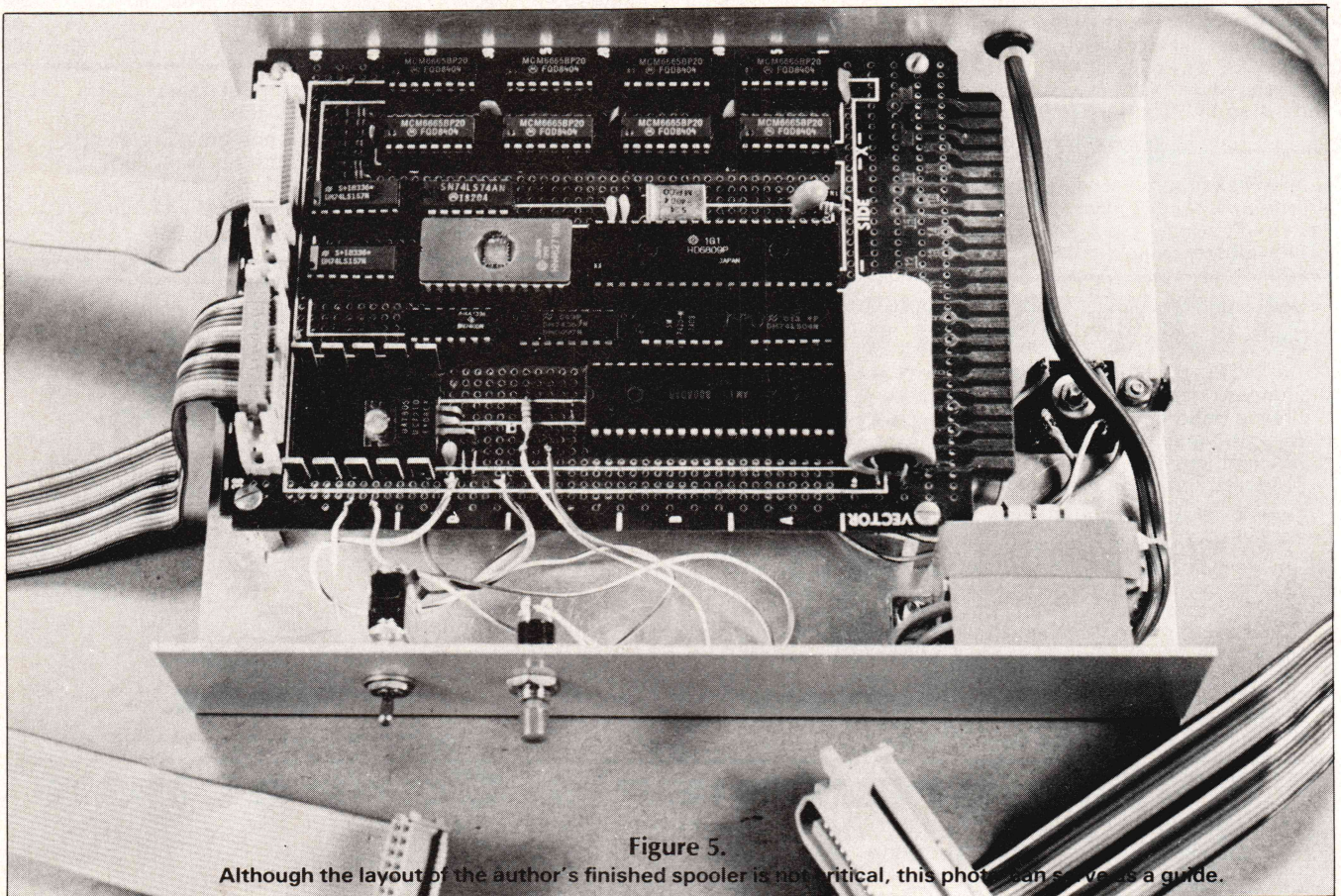


Figure 5.
Although the layout of the author's finished spooler is not critical, this photo can serve as a guide.

(1) The software must read every possible combination of the seven low-order addresses (A0-A6), at least one every 2 ms, to refresh the dynamic RAM.

(2) The software must bank-switch between the upper and lower halves of the ROM to provide the pause function.

Aside from this, the software is straightforward. The 6809 supports relocatable programs with its "long branch" instructions. Listing One (page 53) shows how these functions are implemented.

After a short routine to initialize the PIA, the program wakes up the printer with a null and sets up the X, Y, and U registers as pointers and counters. It next enters a loop that starts with a RAM refresh, which consists of stepping through 128 consecutive ADDA #0 instructions; this steps through our 128 consecutive addresses. The Motorola literature recommends 128 NOPs, which execute in 256 μ s, but I have elected to use 64 consecutive addresses in 128 μ s. During this sequence, once each byte, RAS on the dynamic RAMs is activated. Notice that the eight low-order addresses correspond to the 128 rows in the dynamic RAMs even though we are executing code reading the ROM. CAS remains high, and because the RAMs require both RAS and CAS for a read or write, the RAMs do not assert any data on the data bus. Refresh of the RAMs, however, does take place because the RAS signal alone sets off the refresh internally.

After the refresh sequence, the actual input, storage, and output of characters take place. It is critical that this routine not use any secondary loops that might prevent refresh. For example, if the printer is not ready, the routine continues on rather than waiting in a tight loop for it to get ready.

To implement a pause function with minimum hardware, we have put the same code in both halves of the ROM except for one instruction (see Listing Two, page 55): a BRA (branch always) operation in the active half of the ROM and a BRN (branch never) operation in the pause half. The change of a single bit (\$20

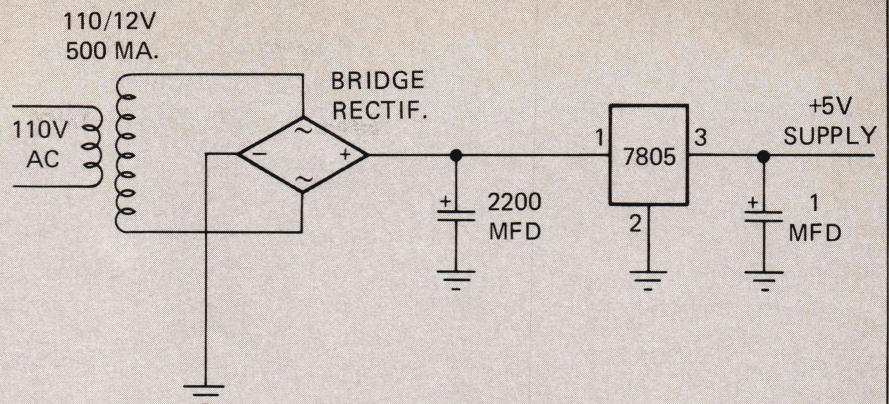


Figure 6.
The rectifiers, transformer, and switches are mounted in the enclosure, while the rest of the power-supply components fit on the circuit board (see Figure 5).

CP/M-80 C Programmers . . .

Save time

. . . with the BDS C Compiler. Compile, link and execute *faster* than you ever thought possible!

If you're a C language programmer whose patience is wearing thin, who wants to spend your valuable time *programming* instead of twiddling your thumbs waiting for slow compilers, who just wants to work *fast*, then it's

time you programmed with the BDS C Compiler.

BDS C is designed for CP/M-80 and provides users with quick, clean software development with emphasis on systems programming.

BDS C features include:

- Ultra-fast compilation, linkage and execution that produce directly executable 8080/Z80 CPM command files.
- A comprehensive debugger that traces program execution and interactively displays both local and external variables by name and proper type.
- Dynamic overlays that allow for run-time segmentation of programs too large to fit into memory.
- A 120-function library written in both C and assembly language with full source code.
- Plus . . .
- A thorough, easy-to-read, 181-page user's manual complete with tutorials, hints, error messages and an easy-to-use index — it's the perfect manual for the beginner and the seasoned professional.
- An attractive selection of sample programs, including MODEM-compatible telecommunications, CPM system utilities, games and more.
- A nationwide BDS C User's Group (\$10 membership fee — application included with package) that offers a newsletter, BDS C updates and access to public domain C utilities.

Reviewers everywhere have praised BDS C for its elegant operation and optimal use of CP/M resources. Above all, BDS C has been hailed for its remarkable speed.

BYTE Magazine placed BDS C ahead of all other 8080/Z80 C compilers tested for fastest object-code execution with all available speed-up options in use. In addition, BDS C's speed of compilation was almost *twice* as

fast as its closet competitor (benchmark for this test was the Sieve of Eratosthenes).

"I recommend both the language and the implementation by BDS very highly."
Tim Pugh, Jr.
in *InfoWorld*
"Performance: Excellent.
Documentation: Excellent.
Ease of Use: Excellent."
InfoWorld
Software Report Card
"... a superior buy ..."
Van Court Hare
in *Lifelines/The Software Magazine*

Don't waste another minute on a slow language processor. Order your BDS C Compiler today!

Complete Package (two 8" SS50 disks, 181-page manual): **\$150**
Free shipping on prepaid orders inside USA.
VISA/MC, COD's, rush orders accepted.
Call for information on other disk formats.

BDS C is designed for use with CP/M-80 operating systems, version 2.2 or higher. It is not currently available for CP/M-86 or MS-DOS.

BDS Software

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

Circle no. 12 on reader service card.

vs. \$21) causes the output portion of the routine to be bypassed when you close the pause switch. See the parts list (page below) for a source for a preprogrammed ROM if you do not care to program your own 2716.

Construction

The only critical aspect of the spooler board's construction is to provide heavy power connections and good bypassing for the dynamic RAM chips. I used the Vector 3677-2 board, which

has wide ground and 5 volt busses. This 4.5 × 6.5-inch board will accommodate the entire circuit, except for the transformer and rectifier, if you position the IC sockets carefully. If you are not sure, use the 4.5 × 9-inch board (Vector 3677).

The RAM bypass capacitors must be close to the power pins of the RAM chips. I used stripped wire-wrap throughout, rather than Just-wrap or equivalent, because my technique with the shortcut methods leaves an

occasional bad connection, which is tedious to debug. Because we do not use the edge-connector, there is space to place the large power-supply capacitor on the board. The only off-board components are the power supply transformer, the rectifier, and the control switches.

Possible Design Modification

You can alter the design for use with a serial printer by replacing the PIA with a 6850 ACIA and adding a suitable baud-rate generator. The ACIA has some built-in handshaking capabilities that can control the interfacing of the host computer and printer.

As to memory refresh, you can divorce the refresh from the mainline routine by interrupting the microprocessor at 500 Hz or faster (2 ms). The interrupt routine would read your 128 consecutive addresses and then either return or do any other short function required before returning from interrupt.

By further decoding, you could increase the buffer size from 60K to about 63.5K, using only 0.5K for the ROM space and only a few bytes for the PIA. Note that we do not use the upper 4K of RAM in our design. Also note that all of the low RAM space is available for character buffering because the software uses internal microprocessor registers exclusively as pointers and counters.

Because the 60K spooler is a "smart" device with its own microprocessor, you could use routines to add a variety of control functions, such as automatic form feed. If your software gets complex, you may wish to implement a stack in RAM and call a subroutine for refresh, e.g.:

LOOP BSR REFRESH

(test a flag)

BPL LOOP

(continue)

Note that the loop contains a call to the refresh routine. Each loop where a significant delay is encountered should contain such a call.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 193.

Parts List

Integrated Circuits

- 1 MC6809 microprocessor (do not use MC6809E)
- 1 MC6821 peripheral interface adapter
- 8 MCM6665A or 4164 64K dynamic RAM 200 ns
- 2 74LS157 quad 2-line to 1-line data selector
- 1 74LS00 quad NAND gate
- 1 74LS04 hex inverter
- 1 74LS20 dual 4-input NAND gate
- 1 74LS367 hex bus driver
- 1 74LS74 dual D-type flip-flop
- 1 2716 EPROM, 5-volt type. Programmed version available at the Bit Stop, 5958 Shenandoah Rd., Mobile, AL 36608 (\$30 ppd)
- 1 7805 5-volt regulator TO-220 case

Sockets and Plugs

- 2 40-pin wire-wrap sockets
- 1 24-pin wire-wrap sockets
- 11 16-pin wire-wrap sockets
- 4 14-pin wire-wrap sockets
- 2 20-pin wire-wrap header, JDR Microdevices IDH20W or equivalent
- 2 20-contact ribbon header sockets, JDR IDS20 or equivalent, plus 1–2 feet 20-conductor ribbon cable

Miscellaneous

- 1 Vector 3677-2 4.5 × 6.5-inch board or Vector 3677 4.5 × 9.0-inch board
- 1 4.0 MHz crystal or TV color-burst crystal
- 1 Heat sink for TO-220 regulator
- 1 1-amp 50 v. bridge rectifier
- 1 Power transformer, primary 110 v. secondary 12 v. at 500 ma
- 1 Reset pushbutton, normally open
- 1 Pause switch, SPST
- 1 2200 mfd 15 v. electrolytic cap.
- 1 33 mfd 15 v. tantalum cap.
- 2 27 pf. disc cap.
- 8 .05 mfd disc cap.
- 3 2 mfd 15 v. tantalum cap.
- 1 1N4001 diode
- 8 33 ohm ¼ w. resistor
- 1 56K ½ w. resistor
- 1 6 × 8 × 2-inch aluminum enclosure or larger (Radio Shack)

Most of the required parts are available from these alternate sources:

JDR Microdevices, 1224 S. Bascom Ave., San Jose, CA 95128
ordering line 1-800-538-5000; in CA 1-800-662-6279

Priority One Electronics, 9161 Deering Ave., Chatsworth, CA 91311
ordering line 1-800-423-5922

DoKay Computer Products, 2100 De La Cruz Blvd., Santa Clara, CA 95050
ordering line 1-800-538-8800; in CA 1-800-848-8008

Listing One

An assembled listing of the ROM contents. The 2K EPROM occupies locations \$F8000 to \$FFFF.

```

1000 * SPOOLER64 - DUAL
1010      .OR $F800
1020      .TA $0800
1030 *      .TF SPOOLER64.OBJ5
1040 STACK .EQ $F3FF
1050 PORTA .EQ $F400
1060 DDRA  .EQ $F400
1070 CRA   .EQ $F401
1080 PORTB .EQ $F402
1090 DDRB  .EQ $F402
1100 CRB   .EQ $F403
1110 SOM   .EQ $0000
1120 EOM   .EQ $F000
1130 MAX   .EQ EOM-SOM
1140 *-----
F800-      1150      .BS $FB00-* SKIP $0300 BYTES
1160 *-----
FB00- 7F F4 01 1170 START CLR CRA      ACCESS DDR'S
FB03- 7F F4 03 1180      CLR CRB
FB06- 86 00      1190      LDA #$00      A - INPUT
FB08- B7 F4 00 1200      STA DDRA
FB0B- 86 FF      1210      LDA #$FF      B - OUTPUT
FB0D- B7 F4 02 1220      STA DDRB
FB10- 86 2F      1230      LDA #$2F      00101111
FB12- B7 F4 01 1240      STA CRA
FB15- 86 2F      1250      LDA #$2F      00101111
FB17- B7 F4 03 1260      STA CRB
FB1A- 86 00      1270      LDA #$00
FB1C- B7 F4 02 1280      STA PORTB      TOSS NULL
FB1F- B6 F4 00 1290      LDA PORTA      STROBE CA2
FB22- 8E 00 00 1300      LDX #SOM      X = INPTR
FB25- 10 8E 00
FB28- 00      1310      LDY #SOM      Y = OUTPTR
FB29- CE 00 00 1320      LDU #0      U = CHRCNT
FB2C- 10 CE F3
FB2F- FF      1330      LDS #STACK
1340 *-----
1350 * REFRSH IS HERE. CONSISTS OF
1360 * 64 ADDA #0 CODES FOR REFRESH.
1370 * INSTRUCTION IS 8B 00
1480 *-----
FBB0- 11 83 F0      1490 CHKIN  CMPU #MAX      CHECK COUNT
FBB3- 00      1500      BEQ PAUSE      SKIP IF MAX.
FBB4- 27 14      1510      LDB CRA      GET INPUT FLAG
FBB6- F6 F4 01 1520      BPL PAUSE      SKIP IF CLEAR
FBB9- 2A 0F      1530      LDA PORTA      GET INPUT DATA
FBBB- B6 F4 00 1540      STA ,X+      STORE & INCR X
FBBE- A7 80      1550      LEAU 1,U      INCR. COUNT
FBC0- 33 41      1560      CMPX #EOM      END OF MEM?
FBC2- 8C F0 00 1570      BNE PAUSE      BRANCH IF NOT
FBC5- 26 03      1580      LDX #SOM      WRAPAROUND
FBC7- 8E 00 00 1590 PAUSE  BRN CHKOUT      NEVER BRANCH
FBCA- 21 03      1600      LBRA REFRSH
FBCF- 11 83 00
FBD2- 00      1610 CHKOUT  CMPU #0      COUNT ZERO?
FBD3- 27 19      1620      BEQ GOREFR      SKIP IF 0
FBD5- F6 F4 03 1630      LDB CRB      GET PRTR READY
FBD8- 2A 14      1640      BPL GOREFR      BRANCH IF BUSY
FBD A- A6 A0      1650      LDA ,Y+      GET DATA. INCR Y
FBD C- B7 F4 02 1660      STA PORTB      PRINT IT
FBD F- B6 F4 02 1670      LDA PORTB      CLEAR IRQ FLAG
FBE2- 33 5F      1680      LEAU -1,U      DECR COUNT
FBE4- 10 8C F0
FBE7- 00      1690      CMPY #EOM      AT MEMORY END?
    
```

(Continued on next page)

JDR DELIVERS QUALITY FOR LESS!

MODEM FOR APPLE OR IBM

\$69.95 SPECIFY APPLE OR IBM

FCC APPROVED

- * BELL SYSTEMS 103 COMPATIBLE
- * 300 BAUD
- * AUTO-DIAL/AUTO-ANSWER
- * DIRECT CONNECT
- * INCLUDES ASCII PRO-EZ™ MENU DRIVEN SOFTWARE
- * INCLUDES AC ADAPTOR

DISKETTE FILE

\$8.95

WITH PURCHASE OF 50 DISKETTES
\$9.95 IF PURCHASED ALONE
HOLDS 70 5 1/4" DISKETTES



NASHUA DISKETTES

5 1/4" SOFT SECTOR,

DS/DD BULK PACKAGED
\$.89ea QTY 250 \$.95ea QTY 100 \$.99ea QTY 50

ORDER TOLL FREE
800-538-5000
800-662-6279 (CA)

TAXAN RGB VISION III

SUPER HI-RES RGB MONITOR

ORIGINALLY MADE FOR ACORN COMPUTER

- * 12" SCREEN
- * 640 x 262 PIXELS
- * SAME SPECS AS MODEL 420

NO C.O.D. ORDERS PLEASE

4164-200 9/\$10.50
8087-3 \$129.00

DISK DRIVES

FOR IBM

TEAC FD-55B \$89.95
HALF HEIGHT DS/DD

MPI-B52 \$89.95
FULL HEIGHT DS/DD

TANDON TM100-2 \$99.95
FULL HEIGHT DS/DD

FOR APPLE

BAL-525
\$119.95



- * 1/2" HEIGHT ALPS MECHANISM
- * 100% APPLE COMPATIBLE
- * ONE YEAR WARRANTY

BAL-500
\$139.95



- * TEAC — 1/2" HEIGHT DIRECT DRIVE
- * 100% APPLE COMPATIBLE
- * 40 TRACK WHEN USED WITH OPTIONAL CONTROLLER
- * ONE YEAR WARRANTY

CONTROLLER CARD \$49.95
IIC ADAPTOR CABLE \$19.95

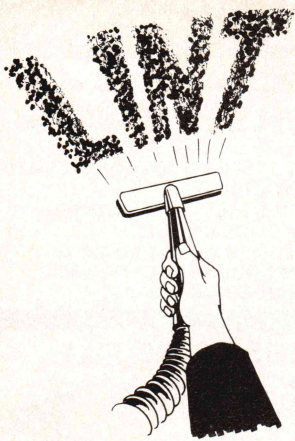
JDR Microdevices

1224 S. Bascom Avenue, San Jose, CA 95128
800-538-5000 • 800-662-6279 (CA) • (408) 995-5430
FAX (408) 275-8415 • Telex 171-110

© Copyright 1985 JDR Microdevices

Circle no. 19 on reader service card.

REMOVE



from your C programs with PC-LINT

PC-LINT analyzes your C programs (one or many modules) and uncovers glitches, bugs, quirks and inconsistencies. It will catch subtle errors before they catch you.

PC-LINT resembles the Lint that runs on the UNIX O.S. but with more features and greater sensitivity to the problems of the 8086 environment.

- Full K&R C
- Supports Multiple Modules—finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or prior to scaling up to a larger memory model.
- All one pass with an integrated pre-processor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- **Introductory Price: \$98.00 MC, VISA**
(includes shipping and handling) PA residents add 6% sales tax. Outside USA add \$10.00.
- Runs on the IBM PC (or XT, AT or compatible) under DOS 2.0 and up, with a minimum of 128KB of memory. It will use all the memory available.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: IBM (IBM Corp.), PC-LINT (Gimpel Software), UNIX (AT&T)

Print Spooler Listing One

(Listing Continued, text begins on page 46)

```

FBE8- 26 04      1700      BNE GOREFR      BRANCH IF NOT
FBEA- 10 8E 00
FBED- 00          1710      LDY #SOM          WRAPAROUND
FBEE- 16 FF 3F    1720 GOREFR LBRA REFRSH      GO TO REFRESH
                        1730 *-----
FBF1-            1740      .BS #FBFE-*        SPACE TO VECTORS
FBFE- FB 00       1750 RESET .DA START        NEED ONLY RESET
                        1760 *-----
                        1770 *-----
FC00-            1780      .BS #FF00-*        SKIP #300 BYTES
                        1790 *-----
FF00- 7F F4 01    1800 STARTX CLR CRA          ACCESS DDR'S
FF03- 7F F4 03    1810      CLR CRB
FF06- 86 00       1820      LDA #00          A - INPUT
FF08- B7 F4 00    1830      STA DDRA
FF0B- 86 FF       1840      LDA #FF          B - OUTPUT
FF0D- B7 F4 02    1850      STA DDRB
FF10- 86 2F       1860      LDA #2F          00101111
FF12- B7 F4 01    1870      STA CRA
FF15- 86 2F       1880      LDA #2F          00101111
FF17- B7 F4 03    1890      STA CRB
FF1A- 86 00       1900      LDA #00
FF1C- B7 F4 02    1910      STA PORTB      TOSS NULL
FF1F- B6 F4 00    1920      LDA PORTA      STROBE CA2
FF22- 8E 00 00    1930      LDX #SOM        X = INPTR
FF25- 10 8E 00
FF28- 00          1940      LDY #SOM        Y = OUTPTR
FF29- CE 00 00    1950      LDU #0          U = CHRCNT
FF2C- 10 CE F3
FF2F- FF          1960      LDS #STACK
                        1970 *-----
                        1980 * REFRSH IS HERE. CONSISTS OF
                        1990 * 64 ADDA #0 CODES FOR REFRESH.
                        2000 * INSTRUCTION IS 8B 00
                        2110 *-----
FFB0- 11 83 F0    2120 CHKINX CMPU #MAX      CHECK COUNT
FFB3- 00          2130      BEQ PAUSEX      SKIP IF MAX.
FFB4- 27 14       2140      LDB CRA          GET INPUT FLAG
FFB6- F6 F4 01    2150      BPL PAUSEX      SKIP IF CLEAR
FFB9- 2A 0F       2160      LDA PORTA      GET INPUT DATA
FFBB- B6 F4 00    2170      STA ,X+        STORE & INCR X
FFBE- A7 80       2180      LEAU 1,U        INCR. COUNT
FFC0- 33 41       2190      CMPX #EOM      END OF MEM?
FFC2- 8C F0 00    2200      BNE PAUSEX      BRANCH IF NOT
FFC5- 26 03       2210      LDX #SOM        WRAPAROUND
FFC7- 8E 00 00    2220 PAUSEX BRA CHKOUX
FFCA- 20 03       2230      LBRA REFRSX
FFCC- 16 FF 61
FFCF- 11 83 00
FFD2- 00          2240 CHKOUX CMPU #0        COUNT ZERO?
FFD3- 27 19       2250      BEQ GOREFX      SKIP IF 0
FFD5- F6 F4 03    2260      LDB CRB          GET PRTR READY
FFD8- 2A 14       2270      BPL GOREFX      BRANCH IF BUSY
FFDA- A6 A0       2280      LDA ,Y+        GET DATA. INCR Y
FFDC- B7 F4 02    2290      STA PORTB      PRINT IT
FFDF- B6 F4 02    2300      LDA PORTB      CLEAR IRQ FLAG
FFE2- 33 5F       2310      LEAU -1,U      DECR COUNT
FFE4- 10 8C F0
FFE7- 00          2320      CMPY #EOM      AT MEMORY END?
FFE8- 26 04       2330      BNE GOREFX      BRANCH IF NOT
FFEA- 10 8E 00
FFED- 00          2340      LDY #SOM        WRAPAROUND
FFEE- 16 FF 3F    2350 GOREFX LBRA REFRSX      GO TO REFRESH
                        2360 *-----
FFF1-            2370      .BS #FFFE-*        SPACE TO VECTORS
FFFE- FF 00       2380 RESETX .DA STARTX      NEED ONLY RESET

```

End Listing One

Listing Two

A hexadecimal dump of the EPROM. Base address for this dump is \$0800 (corresponds to the first byte of the EPROM).

>\$0800.0BFF

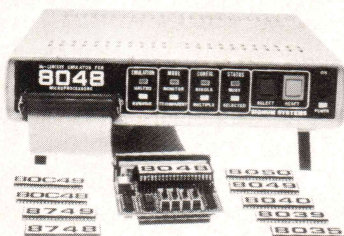
```
0B00- 7F F4 01 7F F4 03 86 00
0B08- B7 F4 00 86 FF B7 F4 02
0B10- 86 2F B7 F4 01 86 2F B7
0B18- F4 03 86 00 B7 F4 02 B6
0B20- F4 00 8E 00 00 10 8E 00
0B28- 00 CE 00 00 10 CE F3 FF
0B30- 8B 00 8B 00 8B 00 8B 00
0B38- 8B 00 8B 00 8B 00 8B 00
0B40- 8B 00 8B 00 8B 00 8B 00
0B48- 8B 00 8B 00 8B 00 8B 00
0B50- 8B 00 8B 00 8B 00 8B 00
0B58- 8B 00 8B 00 8B 00 8B 00
0B60- 8B 00 8B 00 8B 00 8B 00
0B68- 8B 00 8B 00 8B 00 8B 00
0B70- 8B 00 8B 00 8B 00 8B 00
0B78- 8B 00 8B 00 8B 00 8B 00
0B80- 8B 00 8B 00 8B 00 8B 00
0B88- 8B 00 8B 00 8B 00 8B 00
0B90- 8B 00 8B 00 8B 00 8B 00
0B98- 8B 00 8B 00 8B 00 8B 00
0BA0- 8B 00 8B 00 8B 00 8B 00
0BA8- 8B 00 8B 00 8B 00 8B 00
0BB0- 11 83 F0 00 27 14 F6 F4
0BB8- 01 2A 0F B6 F4 00 A7 80
0BC0- 33 41 8C F0 00 26 03 8E
0BC8- 00 00 21 03 16 FF 61 11
0BD0- 83 00 00 27 19 F6 F4 03
0BD8- 2A 14 A6 A0 B7 F4 02 B6
0BE0- F4 02 33 5F 10 8C F0 00
0BE8- 26 04 10 8E 00 00 16 FF
0BF0- 3F 00 00 00 00 00 00 00
0BF8- 00 00 00 00 00 00 FB 00
```

>\$0F00.0FFF

```
0F00- 7F F4 01 7F F4 03 86 00
0F08- B7 F4 00 86 FF B7 F4 02
0F10- 86 2F B7 F4 01 86 2F B7
0F18- F4 03 86 00 B7 F4 02 B6
0F20- F4 00 8E 00 00 10 8E 00
0F28- 00 CE 00 00 10 CE F3 FF
0F30- 8B 00 8B 00 8B 00 8B 00
0F38- 8B 00 8B 00 8B 00 8B 00
0F40- 8B 00 8B 00 8B 00 8B 00
0F48- 8B 00 8B 00 8B 00 8B 00
0F50- 8B 00 8B 00 8B 00 8B 00
0F58- 8B 00 8B 00 8B 00 8B 00
0F60- 8B 00 8B 00 8B 00 8B 00
0F68- 8B 00 8B 00 8B 00 8B 00
0F70- 8B 00 8B 00 8B 00 8B 00
0F78- 8B 00 8B 00 8B 00 8B 00
0F80- 8B 00 8B 00 8B 00 8B 00
0F88- 8B 00 8B 00 8B 00 8B 00
0F90- 8B 00 8B 00 8B 00 8B 00
0F98- 8B 00 8B 00 8B 00 8B 00
0FA0- 8B 00 8B 00 8B 00 8B 00
0FA8- 8B 00 8B 00 8B 00 8B 00
0FB0- 11 83 F0 00 27 14 F6 F4
0FB8- 01 2A 0F B6 F4 00 A7 80
0FC0- 33 41 8C F0 00 26 03 8E
0FC8- 00 00 20 03 16 FF 61 11
0FD0- 83 00 00 27 19 F6 F4 03
0FD8- 2A 14 A6 A0 B7 F4 02 B6
0FE0- F4 02 33 5F 10 8C F0 00
0FE8- 26 04 10 8E 00 00 16 FF
0FF0- 3F 00 00 00 00 00 00 00
0FF8- 00 00 00 00 00 00 FF 00
```

End Listing Two

IN-CIRCUIT EMULATOR For 8050, 8049 and 8048 μ C's



For Hardware/Software Design the E232-48 replaces the target microcomputer for complete emulation under control of the host computer. It emulates all of the above μ C's plus their ROM-less and CMOS versions. Its features are:

- ★ Real time emulation up to 11 Mhz.
- ★ Full 4K Emulation Memory.
- ★ Hardware Breakpoints.
- ★ In-line Assembler and Disassembler.
- ★ Upload, Download & Terminal mode drivers for IBM-PC, CP/M-80 and CP/M-86 are included.

Cross Assemblers for 8048 series and other μ P's running on the above host systems-From \$150
E232-48 in-circuit emulator..... \$1795

SIGNUM SYSTEMS

726 Santa Monica Blvd.
Santa Monica, CA 90401 (213) 451-5382

Circle no. 52 on reader service card.

Disk Sale

Dysan

CORPORATION

TYPE	BOX OF 10
5"-SS/DD-48 TPI	19.50
5"-DS/DD-48 TPI	25.50
5"-SS/DD-96 TPI	29.50
5"-DS/DD-96 TPI	37.50
5"-DS/DD-IBM/AT	52.95
8"-SS/SD-48 TPI	23.95
8"-SS/DD-48 TPI	25.50
8"-DS/DD-48 TPI	29.95

Available Soft or Hard Sector
For Plastic Case Add 1.25/Box
Plus Tax & Shipping
- Cash, Visa, Mastercard, COD -

Integral Systems Corp.
2900-H Longmire Drive
College Station, TX 77840
(409) 764-8017

Circle no. 23 on reader service card.

No source code for
your REL files?

REL/MAC

converts a REL file in the Microsoft™ M80 format to an 8080 or ZILOG™ Z80 source code MAC file with insertion of all public and external symbols.

- REL/MAC makes MAC source files
- REL/MOD lists library modules
- REL/VUE displays the bit stream
- REL/PAK includes all of the above
- 8080 REL/MAC demo disk \$10.00

REL/PAK for 8080 only \$99.95
REL/PAK for Z80 & 8080 \$134.95
on 8"SSSD disk for CP/M™ 2.2

Send check, VISA, MC or C.O.D. to

MICROSMITH

COMPUTER TECHNOLOGY

P.O. BOX 1473 ELKHART, IN 46515

1-800-622-4070
(Illinois only 1-800-942-7317)

Circle no. 41 on reader service card.

Adapted by permission of the publisher, Prentice-Hall, Inc., from the forthcoming book, Designing and Troubleshooting a 68000 Microcomputer System, by Alan D. Wilcox, available 1986. No part of this adaptation may be reproduced, in any form or by any means, without written permission from the publisher.

Designing a Real-Time Clock for the S-100 Bus

by Alan D. Wilcox

The author, with time on his hands and mind, builds a hardware clock for his CompuPro 816/Z.

After revising my latest program and printing it out, I set it aside to look over later. Naturally, it was in a stack of earlier versions of the program that didn't quite work right and looked almost the same except for a few hard-to-find bugs. You know the rest of the story: when I came back in a few weeks, I couldn't tell which was the latest and therefore correct copy of my program. If I had jotted down the time or at least the date on my latest copy, I could have identified it then. So I decided it was time to design a clock circuit to do the writing for me automatically.

The result is a clock circuit using the popular National MM58167A real-time clock IC. The circuit requires no changes in the operating system and has battery backup so you need not reset it after turning off the computer. Designed to meet S-100/IEEE-696, it performs easily with virtually any S-100 microprocessor system. Two assembly language programs set and print the time and date; a third program prints the time, date, and filename as a title header then prints out the file in 60-line pages.

Background

There are two basic ways to put a clock in a computer: either modify the system software to keep track of the time or provide a piece of hardware that will keep time independently of the computer. The software approach is easier in that you need not tamper with the computer hardware, but it has several drawbacks. First, when power is off, the clock is off; this requires resetting the clock every time you turn the computer on. Second, you must modify the *system* software to support a time-keeping program; procedures for this modification can be almost incomprehensible to the novice. Finally, the subsequent time-keeping isn't accurate anyway; for example, the clock must be off during disk read or write operations. The hardware approach avoids these problems.

The concept of a hardware clock is simply to have a clock available for the computer to read when it wants the time and date. The clock has its own battery power supply to keep it running when the computer is turned off. The only programming required is for setting the clock and reading it; these straightforward programs do not require an intimate understanding of the system software. Finally, accuracy is crystal controlled and is unaffected by other computer operations.

A number of articles have discussed how to implement a clock in a personal computer. In one design, Calaway and Hill presented what I call a software approach to

Alan D. Wilcox, 60 South Eighth St., Lewisburg, PA 17837.

time-keeping.¹ Their basic idea is to provide a small amount of hardware to generate an interrupt each second; their system responds to this interrupt by updating the total number of pulses counted and thereby updating the clock time. The operating system needs modification, however, which can range from straightforward to profoundly complex.

Hassebrock described a simple approach to keeping the time and date using the Hayes Chronograph.² The Hayes unit is a piece of hardware connected to the computer through a serial data port. When you want the time or date, you run a program that tells the clock to send the information to the computer for printing or display. Because the Chronograph is connected to a serial port, you must configure the port to match the required data rate, and so on. Therefore, you need a modem-type program to access the clock; the program may be either part of the time-setting/reading program or part of the operating system. Hassebrock chose the latter and modified his system somewhat. Overall, the concept of an external serial-data clock is appealing except that a spare serial port may not be available.

A hardware approach described by Ciarcia seemed initially to offer some distinct advantages.³ First, the design uses hardware to keep time: no difficulty keeping time when the power is off, no system software modifications, and no accuracy problems. Second, the clock does not use a serial port, making port initialization unnecessary. The intended application was for use with the Z8-BASIC micro-computer as an intelligent clock. Although the approach has merit, as it stands, it is not a generally useful design.

Design Overview

The design requirement was for a real-time clock to operate in my CompuPro System 816/Z, a Z80B S-100 machine running at 6 MHz. The clock should keep both the time and date, as well as provide a means of interrupting the system either on a regular basis or at a predetermined time. It should maintain an accuracy within 10 sec/month, with or without system power. Except for initially setting the time and date, the operator should not have to interact with the clock in any way.

The clock should meet S-100/IEEE-696 and operate as a slave without affecting normal system operation; likewise, it should require no changes in the system software. Data transfer between clock and system should use I/O-mapped ports. The handshaking should use the S-100 RDY line, and because the processor operates at 6 MHz, some wait states should be available. Clock interrupts to the system should be switchable to any of the S-100 vectored interrupt lines.

Although all initial programs should poll the clock for the time, the clock should be able to use any of the vectored interrupt lines to implement a future real-time multitasking executive; in the long term, this could involve multiprocessing with a new 10 MHz 68000 CPU board in the system. For the near term, however, the software should at least set and display the time, as well as print files with a time/date header.

Hardware Design

The National MM58167A real-time clock IC⁴ provided a solution to the design requirements. As shown in the block diagram (Figure 1, page 58), you can partition hardware using this clock IC into several major modules that include the clock, address decoder, data-bus interface, interrupt switches, and power supply. Once you divide the hardware into modules, the design can proceed in much the same way as in software development: top down, bottom up, or most critical first.

The appropriate design method in this case is to consider the most critical section first. Everything depends on the clock IC, and its requirements come before all else. The block diagram of the MM58167A (Figure 2, page 59) shows that the IC uses five address lines (32 different addresses) and a chip select and has a single bidirectional data bus with separate read/write controls, two interrupt outputs, and a power-down control. Designing to accommodate each of these requirements individually results in the implementation shown in the circuit schematic (Figure 3, page 60).

I decided to use I/O mapping rather than memory mapping for the clock data transfer. With a maximum of 256 ports, only the lower eight bits of the address bus need decoding, and five of these are decoded internally by the clock IC. For maximum flexibility, I chose to use DIP switches to select the three most significant bits of the I/O address. The output of the decoder is used as a chip select for the clock and by the read/write qualifier circuit.

The S-100 data bus in and out lines are buffered with a pair of 74LS244s connected directly to the clock IC. The buffer for data-in (DI from the processor's viewpoint) is strobed using the I/O read qualifiers pDBIN and sINP and the chip select CS. The data-out buffer is strobed using pWR*, sOUT, and the chip select CS. During a typical read bus cycle, for example, the system places the proper address on the address bus, asserts sINP, then asserts pDBIN to strobe the DI buffer and the clock RD* input. In a typical write bus cycle, the system puts the address on the bus, asserts sOUT then asserts pWR* to strobe the DO buffer and the clock WR* input.

Timing

A basic read or write bus cycle has three bus states (BS1, BS2, and BS3), each of which takes 167 ns in a 6 MHz system; consequently, a bus read or write normally completes within 500 ns. However, for I/O devices that require substantial access times, you can extend the bus cycle by adding a number of wait states. To determine whether wait states are required during any read or write bus cycle, the bus master samples the RDY line at the rising edge of the system clock in BS2. If the RDY line is low, then a wait state (BSw) is inserted immediately after BS2; if the RDY line is still low one bus state later, yet another BSw is inserted. The addition of wait states continues until RDY finally goes high; at that point, the bus cycle concludes with BS3.

The MM58167A clock requires the addition of wait states because of its slow access time. For a clock read

operation, the maximum specified time from valid address until valid data is 1050 ns, far longer than a normal read bus cycle with no wait states. The read bus cycle on my 6 MHz CompuPro system is shown in Figure 4 (page 62) with approximate times to scale. Asserting pDBIN at the beginning of BS2 leaves no time for the clock to bring the RDY line low. However, all Z80 microprocessors automatically insert a single wait state for I/O instructions (by carrying out a microcoded instruction in the Z80), so the clock really has until the beginning of the first wait state BSw (auto) to get RDY low.

The timing question becomes: "Can clock RD* be pulled low and can the clock respond with RDY low—all within 167 ns?" The specs for the clock indicate that read-strobe to ready-strobe (clock pin 4) is 150 ns maximum, but the 7406 easily could add another 40 ns in propagation delay to the system RDY line. Add these up. In the worst case, the clock board cannot get the RDY line down in time even by taking advantage of the Z80 automatic wait state. As indicated in Figure 4, the clock actually takes from about 100 ns (middle of BS2) to about 190 ns (middle of BSw). Actual timing measurements and

read operations indicate that the clock sometimes is fast enough but usually is not. For reliable operation of RDY, we must add an extra wait state.

In my system, I used the Z80 processor board's option switch to add a single wait state to I/O operations in addition to the Z80 automatic wait. This required no additional hardware, although you could enhance a more general clock design by including an on-board wait state generator circuit. Libes and Garetz present a number of suitable circuits in their book.⁶

When the clock data is finally valid, the system allows the RDY line to go high, and the processor reads the data in BS3. To finish BS3, pDBIN is negated, which raises the clock RD* input, and the address is deselected.

The write bus cycle timing requirements for the clock are similar to the read timing requirements. As in the read, you must include a single wait state to allow sufficient time for RDY to be pulled low during the write operation.

Power-Down Design

The power-down design is critical in the actual operation of the clock. If POWER DOWN* is not asserted to a low-

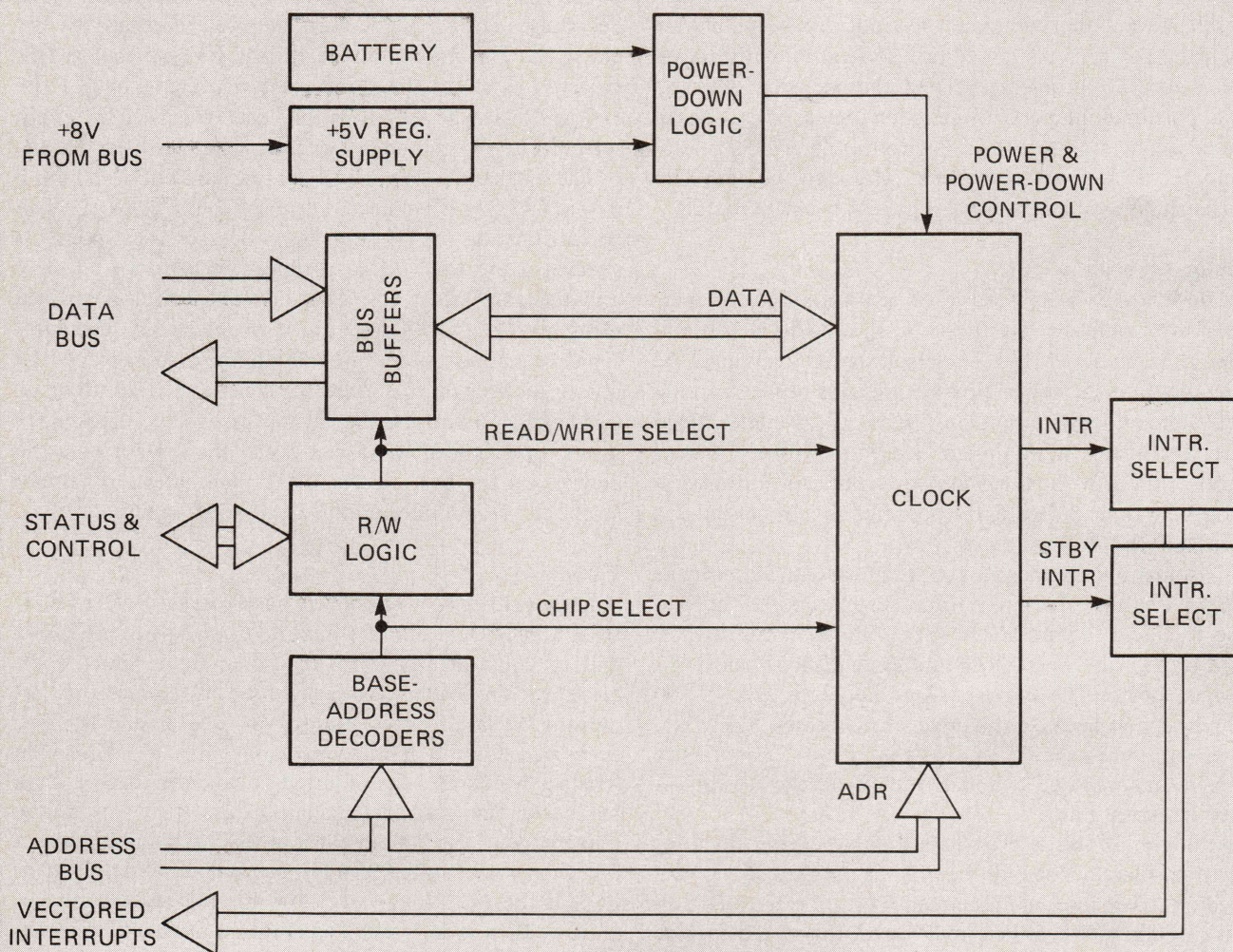


Figure 1.
Block Diagram of Real-Time Clock Board

voltage level at least one microsecond before system power removal, then the contents of the clock memory might be anything (or nothing) when you restore normal system power. Furthermore, when the power does come back on, POWER DOWN* must be held low until all bus signals are valid.

A common way of accomplishing this is to use a zener diode that sets a reference threshold voltage to control a transistor switch. Consider the circuit (Figure 3) when the system power is off: Q1 and Q2 are both off. When the system power comes on, nothing happens until the system bus voltage reaches about 6.6 V. Then the zener conducts, turning on Q2, which then turns on Q1. When Q1 goes on, voltage is applied to POWER DOWN*, which activates the clock chip for normal operation. Because a system bus voltage greater than 6.6 V is enough to allow near-normal output from all 5 V regulators, all bus signals should be valid by this time. When the system power goes off, the system voltage drops gradually enough so that, when it passes below 6.6 V, the zener and Q1/Q2 can drop the POWER DOWN* to a low voltage before the bus signals are no longer valid.

A small, 3.75 V, 20 mAh, NiCad rechargeable battery maintains clock operation when system power is off. Specified current consumption of the clock is on the order of 10 to 20 uA during power-down; my clock measured 14 uA while using the battery with a series-blocking diode. The 470 ohm resistor provides a trickle charge from 1/2

mA to about 2 mA, depending on the state of the battery and component tolerances. For an average use of several hours a day, this resistor keeps the battery charged between 3.8 and 4.15 V with no difficulties.

Logic Circuits

IEEE Std-696 (sec. 3.7) states that a card may not source more than 0.5 mA at 0.5 V nor sink more than 80 uA at 2.4 V on most system signal lines.⁵ The effect of this is that you can connect only one LS TTL gate per card to each line of the address bus. My decoder uses the 74LS136 exclusive-or, which sources a worst-case maximum of 0.8 mA at 0.4 V. I expect this to cause no operational problems in any practical sense. I opted to use the 74LS136, rather than buffer the address bus with extra gates, to be strictly within the specification. I prefer to be conscious of trade-offs than to not be aware of them and have a problem occur later.

The pull-up resistors for the open-collector 7406 are selected to maintain proper logic levels and currents. The design trade-off is to let the various resistors be high for lower current consumption or to let them be low for more speed. I designed for as much speed as possible. Typical calculations are discussed in the standard *TTL Data Book* (page 6-6).⁷

Near the end of the project, I decided to implement the clock standby interrupt, which required an extra non-inverting gate. I used the spare 74LS10 and the last spare

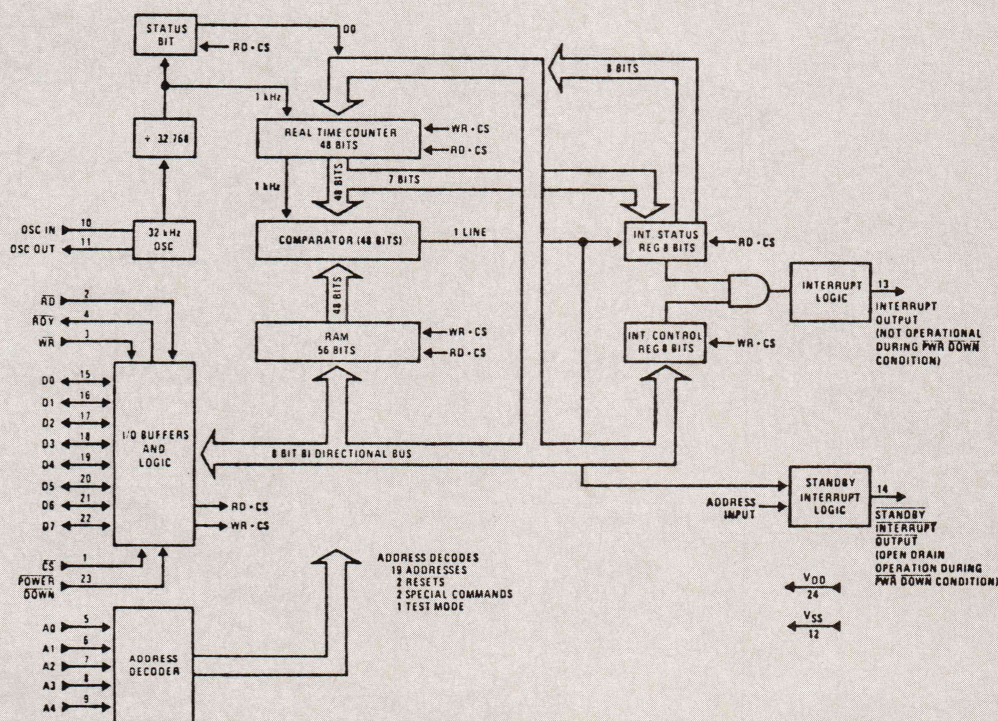
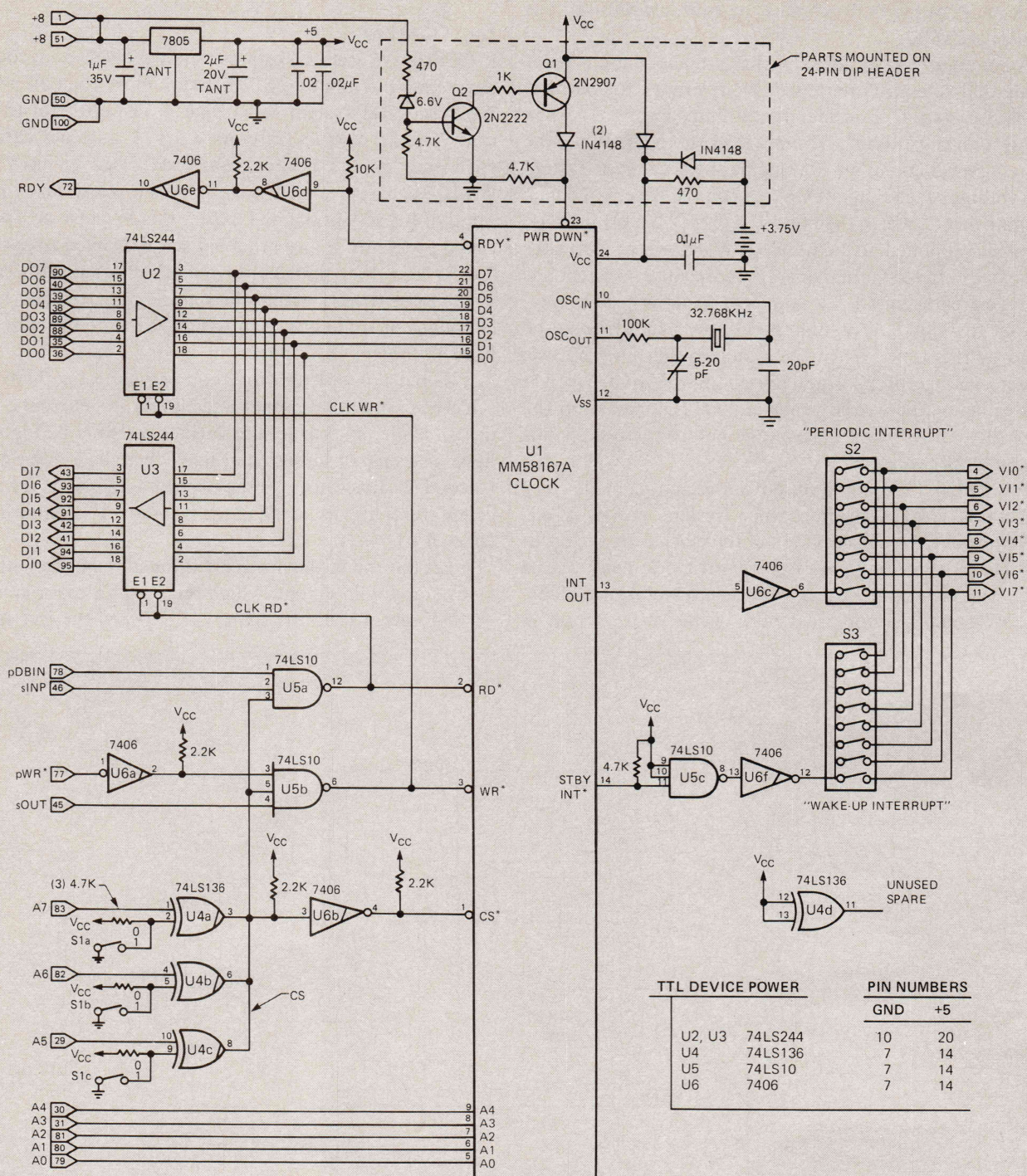


Figure 2.
Block Diagram of the MM58167A (material courtesy of
National Semiconductor Corporation)



*NOTE: Leave interrupt switches S2 and S3 all open unless interrupts are to be implemented. If interrupts are used, close only 1 switch in S2 and 1 switch in S3 at most.

Figure 3.
Schematic Wiring Diagram of S-100
Real-Time Clock

DE SMET C

8086/8088 Development Package

\$109

FULL DEVELOPMENT PACKAGE

- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full Screen Editor
- Execution Profiler
- Complete STDIO Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 & S/W FLOATING POINT

OVERLAYS

OUTSTANDING PERFORMANCE

- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER \$50

- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT \$35

- Converts DeSmet.O to DOS.OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

CW ARE

CORPORATION

P.O. Box C, Sunnyvale, CA 94087

(408) 720-9696

Street Address: 505 W. Olive, #767 (94086) Call for hrs.

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add \$5, elsewhere add \$15. Checks must be on U.S. Bank and in U.S. Dollars. Call 9am-1pm to CHARGE by VISA/MC/AMEX.

Foreign Distributors: AFRICA: HI-TECH SVCS, Gaborone 4540 or Telex 2205BD LANGER • ENGLAND: MLH Tech, 0606-891146 • JAPAN: JSE 03-486-7151 • SWEDEN: ESCORT DATA 08-87 41 48 or THESEUS KONSULT 08-23 61 60

Circle no. 18 on reader service card.

ConIX™

NOW ONLY \$79.95!

If you think you're missing out on innovative software developments because nobody is writing for CP/M™-80, take a look at us. We've adapted UNIX™ features to CP/M like never before, and with the kind of professional, quality-controlled product that you deserve. That product is none other than the critically acclaimed ConIX Operating System.

ConIX can provide any 48K+ CP/M-80 or compatible system with I/O Redirection and Pipes (uses memory or disk), perfected User Areas, Command and Overlay Path Searching, Auto Screen Paging, 8Mb Print Buffering, 22 new SysCalls, Function Keys, "Virtual" disk system, Archiver (saves over 50% disk), extensive command language, 300+ variables, 100+ commands, pull-down menu, and much more! Uses as little as 1/2K RAM! Runs with CP/M for true data and software compatibility. Installs easily without any system mods!

The ConIX package lists at \$165 and has been advertised and sold internationally to many enthusiastic customers since October 1983. As a special limited offer, we've lowered the price of the complete ConIX system by 50% to only \$79.95! Don't miss this opportunity to bring your 8-bit micro back into the software revolution. Order your copy of ConIX today!

Price includes manual, 8" disk, and user support. 5¼" conversions available. Contact your local dealer, or buy direct and add shipping: \$4.50 UPS, \$10 Canada, \$25 overseas. NY residents add sales tax.



Computer Helper Industries Inc.
P.O. Box 680 Parkchester Station, NY 10462
Tel. (212) 652-1786 (for information/orders)

"We're helping your computer work better for you!"

UNIX: AT&T Bell Labs, CP/M: Digital Research, ConIX: Computer Helper Ind.

Circle no. 22 on reader service card.

THE SYMBOLIC SINE QUA NON

L I S P

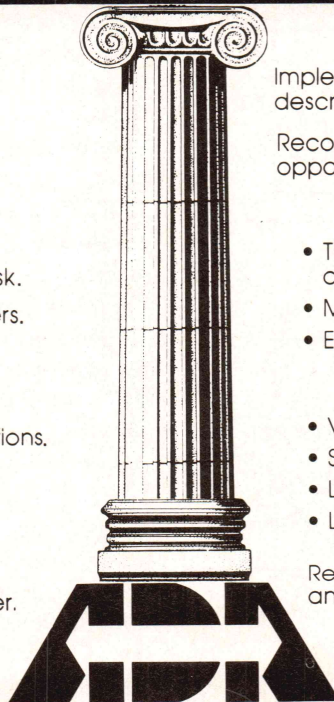
Cybermetrics UNXLISP-86 (tm) Features:

- 1 Megabyte address space.
- LAMBDA, NLAMBDA, MACRO with displacement.
- CATCH, THROW, ERRSET, & ERR.
- SAVE/RESTORE virtual images to/from disk.
- &OPTIONAL, &REST, and &AUX parameters.
- Tree-structured object list, functional directories.
- Lisp utilities with source, including pretty-printer, structure editor, debugging functions.
- Works with any text editor through exec function.
- Extensive manual - written in ENGLISH
- Requires 320K memory and runs under MSDOS and PC DOS versions 2.0 and later.

TECHNICAL
(408) 725-1344

\$49.00

ADD \$3.00 FOR
SHIPPING & HANDLING



automata design assoc.

1570 ARRAN WAY
DRESHER, PA 19025

TO ORDER CALL (215) 355-5400 USE IT FOR 30 DAYS

Circle no. 130 on reader service card.

THE fifth generation language

P R O L O G

Implementing the full Edinburgh Syntax as described by Clocksin and Mellish.

Recognized by Japan as providing unparalleled opportunity for artificial intelligence.

Applications:

- The highest level of a hierarchical robotic control system.
- Machine recognition of natural language.
- Expert systems and knowledge engineering.

Optional:

- Virtual memory
- Special libraries
- Language extensions
- Large model

Requires 192K memory and runs under MSDOS and PC DOS versions 2.0 and later.

TECHNICAL
(215) 646-4894

educational package

\$29.95

other versions \$50-\$500

VISA
MASTERCARD

7406 gate to get this output from the clock board. This particular output does not have the flexibility of the programmable output, but you can use it to indicate a match between real time and a preset time for a wake-up alarm.

Construction and Installation

I prototyped the entire real-time clock on a Vector 8800V wire-wrap board, as shown in the photo (page 64). I added the interrupt switches (S3) for the wake-up interrupt after taking the photo; they are located beside the main interrupt switches (S2). All the circuits took less than a third of a standard S-100 board. I wired the transistor power-down circuit on the 24-pin header at the lower-left edge of the board by the battery. I used plastic "wrap ID" panels on the back of the board for each IC socket to make component and pin identification easier.

My system does not have any I/O port assignments in the range A0h to BFh, so I selected A0 as the base address for the clock. For this base address, S1 sections a and c are closed for a "1" and section b is left open for a "0." As

shown in Figure 5 (page 63), the clock uses a total of 24 addresses rather than the whole block of 32 allocated to it. This permits additional design flexibility if port addresses are at a premium.

You may install the board in any convenient location on the bus. Because of its wire-wrap construction, however, the board takes up the space of two normal boards and cannot be crowded. Depending on your needs, you can snip down the wire-wrap pins considerably (watch your eyes) to make the board thinner.

Software

When writing the programs to set the clock and print the time, I concentrated on writing clear, structured, modular code. Thus, you should have little difficulty making changes to transport the clock software to another system or to make modifications. Naturally, the resulting code perhaps is not as brief as might have been possible were I writing only for myself.

I have presented several programs here for using the clock. The first is CLOCKSET (see Listing One, page 68), a program to set either the day and date, the time, or both. The second is TIME (Listing Two, page 74), a program to display the time and date on the console. This same program will print the time and date plus a short text (such as a filename) to LST: if you include the text with the program invocation. This second feature of TIME is handy when you are using a text editor and need a date on a page about to be printed. The last is TIMELIST (see Listing Three, page 79), a program to print a time/date header with filename then list a text file with 60 lines per page.

Recall that I chose to use A0h as a clock base address for I/O. In all the programs, I used CLOCK as the base address and set its value equal to A0h. I identify other clock port addresses by a simple reference to CLOCK+1, CLOCK+2, and so on. For example, from Figure 5, CLOCK+3 is the address for "minutes" and CLOCK+7 is the address for "months." Using CLOCK in this way allows you to make an easy change to any base address just by changing one statement at the beginning of the program.

The first program, CLOCKSET, asks whether or not to set the date. If you select the default "No," then it asks whether to set the time. If you choose a second default "No," the program terminates without changing the clock. If, however, you do select the date-set option, then it asks for two digits for the day of the week: enter 01 for Sunday, 02 for Monday, and so on. You enter the day of the month the same way: 01 through 31. You enter the month itself as month 01 through 12. Select the year by entering just the last two digits; that is, 85 for 1985. Each January, you must change the year manually; the clock IC does not automatically change to a new year. When writing this program, I opted to keep it simple: no error checks and nothing fancy like entering literal names instead of numbers. If I make an error, I just invoke the program over again and do it correctly the next time through.

The second part of CLOCKSET is similar to the date-setting procedure. You enter the time in two-digit groups for the hours and minutes, and a carriage return sets the

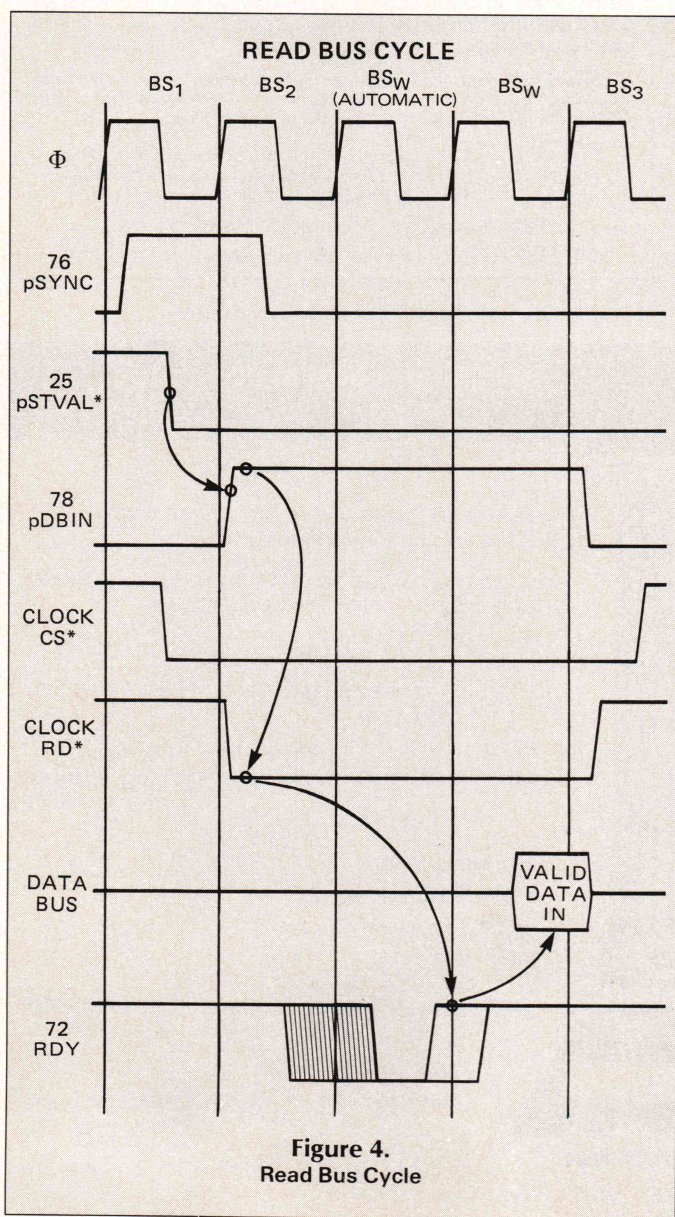


Figure 4.
Read Bus Cycle

Functional Description

TABLE II. ADDRESS CODES AND FUNCTIONS

A4	A3	A2	A1	A0	Function
0	0	0	0	0	Counter—Ten Thousandths of Seconds
0	0	0	0	1	Counter—Hundredths and Tenths of Seconds
0	0	0	1	0	Counter—Seconds
0	0	0	1	1	Counter—Minutes
0	0	1	0	0	Counter—Hours
0	0	1	0	1	Counter—Day of Week
0	0	1	1	0	Counter—Day of Month
0	0	1	1	1	Counter—Month
0	1	0	0	0	RAM—Ten Thousandths of Seconds
0	1	0	0	1	RAM—Hundredths and Tenths of Seconds
0	1	0	1	0	RAM—Seconds
0	1	0	1	1	RAM—Minutes
0	1	1	0	0	RAM—Hours
0	1	1	0	1	RAM—Day of Week
0	1	1	1	0	RAM—Day of Month
0	1	1	1	1	RAM—Months
1	0	0	0	0	Interrupt Status Register
1	0	0	0	1	Interrupt Control Register
1	0	0	1	0	Counters Reset
1	0	0	1	1	RAM Reset
1	0	1	0	0	Status Bit
1	0	1	0	1	GO Command
1	0	1	1	0	STANDBY INTERRUPT
1	1	1	1	1	Test Mode

Figure 5.
Clock Address Codes (material courtesy of
National Semiconductor Corporation)

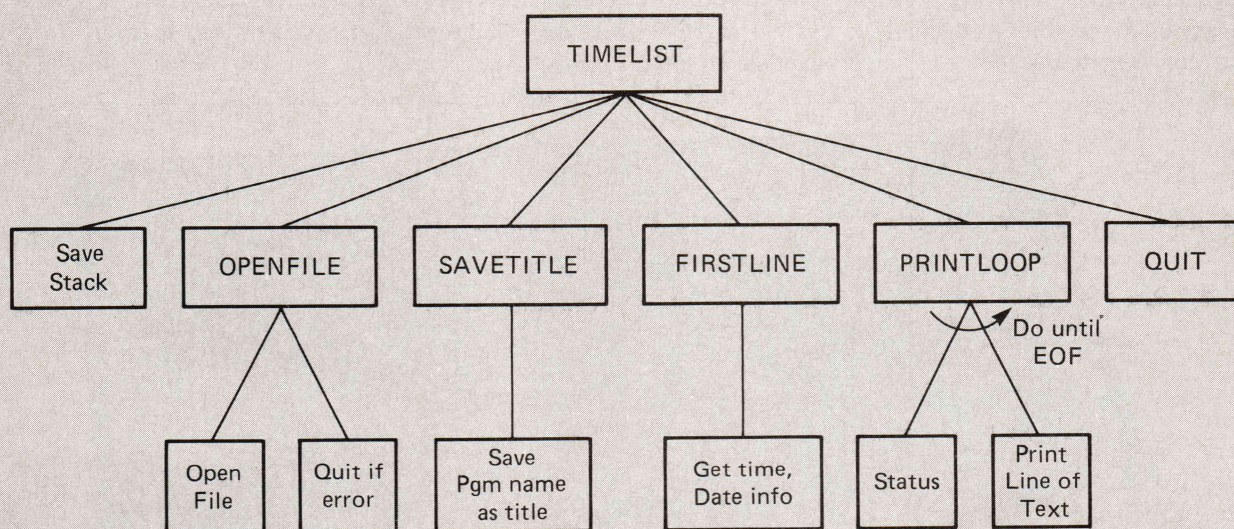


Figure 6
Structure Chart of the TIMELIST Program

seconds to zero. The clock chip keeps its time based on a 24-hour count, so if the time is 7:15 pm, for example, then the entry is 19 hours, 15 minutes. The carriage return to set seconds to zero does a write-data to `CLOCK+21` (the `GO` command, see Figure 5). This write pulse resets the thousandths, hundredths, tenths, units, and tens of seconds counters. In the `SETTIM` subroutine, note that I reset the seconds (an early `GO`) before setting the minutes; this is because the clock chip will increment the minute counter upon `GO` if the seconds counter is greater than 40. For example, if the time is almost 7:15 pm and I enter 19 and 15 followed by `<cr>` to zero seconds, I might find the time erroneously set to 19:16; hence, the need to reset `GO`.

In the program `TIME`, unlike the simpler `CLOCKSET` program, I wanted more than just bare numbers from the clock: I wanted the literal names for days and months. The subroutine `NTOLIT` makes the conversion from the clock's packed BCD to a literal string. It does this by converting the number (say, 11 for November) to binary then indexing through the list of literals until it reaches the eleventh one. Then it moves the word "November" in memory to the space reserved for the message to be printed.

You can use `TIME` in two ways: either as a simple display to the console or as a display plus an output to the printer. To get the display, type `TIME`. To get the printer output in addition to the console display, type `TIME <text>`. The additional letters will send the time and date information plus the `<text>` to the currently defined `LST`: device connected to the computer. This is one way of time/date-stamping a page before sending some special output to the printer. Because the print output is aligned toward the right-hand side of the page, you have space for only about 15 characters before the printer wraps around. I use the space to put in a filename or some other text identification. Another way I use this feature is to print a disk name as `<text>` then send a disk directory to the printer; I keep this directory listing with the disk so I have a dated record of its contents.

For convenient time/date-stamping of a full program

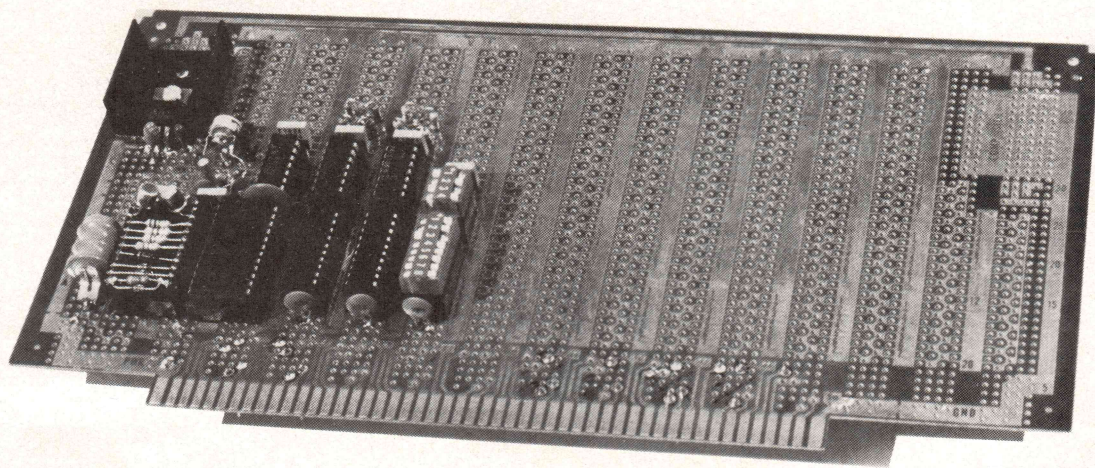
listing, I chose the approach in the structure chart shown in Figure 6 (page 63). This program is invoked by `TIMELIST <filename>`. First, the title line is printed in the format: time, date, program name. Then, after skipping several lines, the printout of the text of the program begins, with a total of 60 lines on each page. All of the printing is done in subroutine `PRINTLOOP`, and you can interrupt it anytime from the keyboard; that is, a control-S pauses and any keypress continues, unless the printing is not paused, in which case any keypress aborts the program.

The clock-related aspects of this program are similar to `TIME`. The obvious difference is that a file must be opened, read, and printed. At the very beginning, then, because I used the 128 bytes between 80H and 100H for a disk buffer area, it is necessary to move the stack to a safe place. Then the first subroutine, `OPENFILE`, either opens the file successfully or calls `QUIT` if it encounters a problem. On exit, the stack pointer is restored to its previous position.

`PRINTLOOP` sends one character for each main loop within the subroutine. Before getting each new character, the program checks to see if a console break is present, to verify required spaces at the beginning of a line, and to determine if the maximum number of lines per page has been reached. The program as now assembled adds three spaces at the left margin for all printed text; this is because I usually notebook-punch my programs and my Okidata 82A prints too closely to the left-hand margin. To change this margin setting, adjust the `SKIP` equate. Likewise, to change from 60 lines per page, modify the `LPAGE` equate.

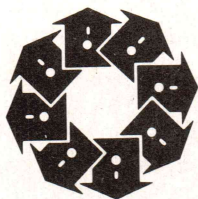
After getting a new byte by a call to `GETNB`, I check for the form-feed character and ignore it. To allow form feeds from the text, change the `KEEPFF` equate to `TRUE`. When accepting form feeds from the text, I disabled the line-counting within `PRINTLOOP` on the assumption that page length would have already been determined.

Before actually printing the character, the program checks for the proper line length. This is necessary because I skip several spaces at the left margin. If the text runs to 80 characters, but I've already printed three



The real-time clock on a Vector 8800 V wire-wrap board.

Does your **ISAM**
run on **IBM,**
APPLE, DEC
and **AT&T**
computers?
c-tree does, and
you only **BUY**
IT ONCE!



c-tree™
BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

The company that introduced micros to B+ Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B+ Tree based file handlers. With c-tree™ you get:

- complete C source code written to K&R standards of portability
- high level, multi-key ISAM routines and low level B+ Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

\$395 COMPLETE

Specify format:
5¼" PC-DOS 3½" Mac
8" CP/M® 8" RT-11

for VISA, MC or COD orders, call
1-314-445-6833

Access Manager and CP/M are trademarks of Digital Research, Inc. Apple is a trademark of Apple Computer, Inc. c-tree and the circular disc logo are trademarks of FairCom. IBM is a trademark of International Business Machines Corporation. DEC is a trademark of Digital Equipment Corporation. ©1984 FairCom.

Circle no. 37 on reader service card.

PC **TEX** is here!

Complete **TEX**82 Typesetting for your PC/XT or AT

- Real, state-of-the-art typesetting capable of handling all mathematical and scientific material.
- Produce work of this quality on your Epson printer.

$$\sum_{p \text{ prime}} f(p) = \int_{t>1} f(t) d\pi(t). \quad \overbrace{\{a, \dots, a, b, \dots, b\}}^{k \text{ a's} \quad l \text{ b's}} \\ k+1 \text{ elements}$$

- Outperform professional typesetters with work of this quality from a QMS Lasergrafix.

$$G(z) = e^{\ln G(z)} = \exp\left(\sum_{k \geq 1} \frac{S_k z^k}{k}\right) = \prod_{k \geq 1} e^{S_k z^k / k}$$

- PCT_{EX} includes INITEX, La_{TEX} macro package and manual, PCT_{EX} macro package and manual.
- PCT_{EX} is a full implementation of Donald Knuth's **TEX** and produces standard .DVI files.
- PCT_{EX}: only \$279. PCDOT dot-matrix printer driver (includes over 200 fonts): \$100. (For IBM Graphics printer, Epson RX FX printers, Toshiba 134x 135x.) QMS Lasergrafix driver: \$200. Include \$3. shipping for each order. Calif. residents add sales tax. MasterCard, Visa accepted. Requires DOS 2.0 or better, 512K RAM, 10M hard disk.

PERSONAL
TEX
INC

20 Sunnyside, Suite H, Mill Valley, CA 94941.
(415) 388-8853. Telex 275611.

TEX: American Mathematical Society. PCT_{EX}: Personal TEX, Inc.. IBM-PC: IBM Corp.. QMS: QMS, Inc.

Circle no. 76 on reader service card.

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support. Full range transcendentals (tan, sin, cos, arctan, logs and exponentials) Data type conversion and I/O formatting.
- High level interrupt support. Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS / FORTH

- Fully Optimized & Tested for:
IBM-PC IBM-XT IBM-JR
COMPAQ EAGLE-PC-2
TANDY 2000 CORONA

LEADING EDGE

(Identical version runs on almost all MSDOS compatibles!)

- Graphics & Text (including windowed scrolling)
- Music - foreground and background includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler (interactive, easy to use & learn)
- Compare

BYTE Sieve Benchmark jan 83
HS/FORTH 47 sec BASIC 2000 sec
w/AUTO-OPT 9 sec Assembler 5 sec
other Forths (mostly 64k) 70-140 sec

FASTEST FORTH SYSTEM AVAILABLE.

TWICE AS FAST AS OTHER
FULL MEGABYTE FORTHS!

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: \$250.

VISA Visa Mastercard
Add \$10. shipping and handling

HARVARD SOFTWORKS

PO Box 69
Springboro, Ohio 45066
(513) 748-0390

Circle no. 44 on reader service card.

spaces, then three characters will have to wrap around to another line on the printer. To maintain the left margin, I need to insert the usual spaces before beginning to print the wrap-around letters.

If the character to be printed is a tab character (09), then TABTOSP does a conversion to spaces so that the printer lines up to every eighth column. The conversion is *not* simply to replace every tab with eight spaces: the program checks to see what the current column position is in relation to every eighth column then inserts just enough spaces to get the next eighth position. Thus, a tab positions the printer to columns 9, 17, 25, 33, and so on. Change the NRSPCS equate to the desired tab conversion if eight is not suitable.

As with all programs, a number of modifications always seem necessary. The CLOCKSET, TIME, and TIME-LIST programs are certainly not the final versions. Improvements are always possible; it becomes an issue of whether the programs are "good enough to do the job" or *must* be revised. I chose to leave these three programs as they stand and go on to other more creative programs.

One program is to display the current time on the console, but I have a wristwatch that displays time quite well. Consider a "reminder" program to sound the console bell at a certain time. Also consider how to keep tax records of business use of the computer: a program to LOGON and LOGOFF that records date, time on/off, and subject of computer use. This program should create a file suitable for use by a spreadsheet program for tax records. These programs and others using the interrupt features of the clock are the subject of a future article.

Operation

When first starting up the clock, set the proper time and date using CLOCKSET. Then run the TIME program to see that the time and date are being displayed properly. Also check the printing feature of TIME by appending a small text to the TIME invocation to see that the time and text go to the printer. Test run TIMELIST with a two- or three-page text file to see that it prints the first page with time and date then prints 60 lines on each of the following pages.

Conclusions

The clock board has proven to be a useful addition to my computer system. I've found it especially convenient for dating disk directory prints, as well as for the normal dating of program printouts. Keeping the time and date on my program listings has certainly improved my program filing system: now when I discover a long-lost paper, I know when it disappeared!

I would like to thank our Bucknell Technician, Thomas J. Thul, Jr., for his help in laying out and constructing the clock circuit board. Program source code is available from the author on an 8-inch CP/M disk for \$15 postpaid.

References

- ¹ Calaway, J. L., and B. Hill, "CP/M, Your Time Has Come," *BYTE*, May 1982, pp. 479-493.

- ² Hassebrock, Glen E., Jr., "The Hayes Chronograph, an H-89, and CP/M," *REMark*, No. 35, December 1982, pp. 9-14.

- ³ Ciarcia, Steve, "Everyone Can Know the Real Time," *BYTE*, May 1982, pp. 34-58.

- ⁴ *MM58167A Microprocessor Real Time Clock*, National Semiconductor Data Sheet, April 1982.

- ⁵ *IEEE Standard 696 Interface Devices*, New York, NY: IEEE, 1983.

- ⁶ Libes, Sol, and Mark Garetz, *Interfacing to S-100/IEEE-696 Microcomputers*, Berkeley, CA: Osborne/McGraw-Hill, 1981.

- ⁷ *The TTL Data Book for Design Engineers*, 2nd ed., Dallas, TX: Texas Instruments, Inc., 1981.

Trademarks

Hayes Chronograph is a trademark of Hayes Microcomputer Products, and Z8 is a trademark of Zilog.

Hardware: IEEE Standard 696-1983 compatible

Operation to 6 MHz (using two wait states)

Handshaking using RDY line (72) pulled low until clock output data valid

DIP-switch selection of base address on any 32-byte I/O port boundary, 32 ports required

DIP-switch selection of vectored interrupt 0

through 7 for clock interrupts 100 ms

to 1 month or RAM and counter compares

DIP-switch selection of vectored interrupt 0

through 7 for RAM and counter compares

Accuracy 0.01%, but generally better than 5 sec/month

Power requirement +8V < 200 mA during normal operation

Rechargeable battery backup

Construction using Vector 8800V and wire-wrap

Software: 24-hour time: hour, minute, second

Calendar: day of week and month, month, year

File-list utility with time/date header

Clock time- and date-setting program

Table

Specifications of the S-100 Real-Time Clock

DDJ

(Listing begins on page 68)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 194.

Fortran Scientific Subroutine Package

Contains Approx. 100 Fortran Subroutines Covering:

- | | |
|----------------------------------|-----------------------------|
| 1. Matrix Storage and Operations | 7. Time Series |
| 2. Correlation and Regression | 8. Nonparametric Statistics |
| 3. Design Analysis | 9. Distribution Functions |
| 4. Discriminant Analysis | 10. Linear Analysis |
| 5. Factor Analysis | 11. Polynomial Solutions |
| 6. Eigen Analysis | 12. Data Screening |

Sources Included

\$295.00

FORLIB-PLUS™

Contains three assembly coded LIBRARIES plus support. FORTRAN coded subroutines and DEMO programs.

The three LIBRARIES contain support for GRAPHICS, COMMUNICATION, and FILE HANDLING/DISK SUPPORT. An additional feature within the graphics library is the capability of one fortran program calling another and passing data to it. Within the communication library, there are routines which will permit interrupt driven, buffered data to be received. With this capability, 9600 BAUD communication is possible. The file handling library contains all the required software to be DOS 3.0 PATHNAME compatible.

STRINGS & THINGS™

Support for CHARACTER MANIPULATION (string support), SHELL, BATCH, MUSIC, CMD LINE, and ENVIRON CTRL.

\$69.95 each

P.O. Box 2517
Cypress, CA 90630



(714) 894-6808

California residents, please add 6% sales tax.

Versions available for IBM Professional Fortran
or MICROSOFT 3.2 Fortran

Circle no. 1 on reader service card.

Poor Person Software

Introduces

Write-Hand-Man

Desk accessories for CP/M

Write-Hand-Man lets you take notes, check phone numbers, make appointments, and countless other tasks without leaving Wordstar, dBase, Multiplan, or any other application. Enter Write-Hand-Man with a single key-stroke and choose the program you want. When you leave Write-Hand-Man, your application continues normally.

\$49.95 plus tax delivers Write-Hand-Man and 4 companion programs; Notepad, Phonebook, Calendar, and Termcomm. User written programs are easily added. All you need is M80 or some other LINK-80 compatible assembler.

Other CP/M products available from Poor Person Software: Poor Person's Spooler (\$49.95), Poor Person's Spelling Checker (\$29.95), Poor Person's Spread Sheet (\$29.95), Keyed Sequential Files (\$39.95), Poor Person's Menus (\$29.95), aMAZEing Game (\$29.95), Window System (\$29.95), Crossword Game (\$39.95), Mailing Label Processor (\$29.95). Shipping included.

All products available on IBM 8 inch and Northstar 5 inch disks. Other 5 inch formats add \$5 handling charge. No credit cards.

Poor Person Software

3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

Circle no. 71 on reader service card.

2 Megabyte SemiDisk!

Have you been waiting on your slow floppy disk drives too long? SemiDisk Systems has a disk emulator for you! It'll put you in the fast lane, with ultra-fast data transfer, huge storage capacity, convenient battery backup, and a handy print spooler.

Have you been waiting for a SemiDisk big enough to handle your large applications programs, files, and databases - all at once? Your wait is over. SemiDisk Systems is now delivering 2 megabytes of disk storage on a single board!

512k, 1Meg and 2Megabyte SemiDisks are available for S-100 computers, (including the H/Z-100 operating under Z-DOS), IBM PC, XT, & AT, the TRS-80 Models II, 12, & 16, and the Epson QX-10. Once you've tried a SemiDisk you'll know why we say. . .

Someday you'll get a SemiDisk.

Until then, you'll just have to wait.

	512K	1Mbyte	2Mbyte
SemiDisk I, S-100	\$995	\$1795	
SemiDisk II, S-100	\$1295	\$2095	\$2549
IBM PC, XT, AT	\$945	\$1795	\$2499
QX-10, QX-16	\$799		\$2499
TRS-80 II, 12, 16	\$995	\$1795	\$2499
Battery Backup Unit	\$150		

SEMIDISK

SemiDisk Systems, Inc.

P.O. Box GG, Beaverton, Oregon 97075

503-642-3100



Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha, all SemiDisk-equipped computer bulletin boards (300/1200 baud). SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle no. 85 on reader service card.

Listing One

```

; CLOCKSET      Program to set date & time of real-time
;
; Program by    Alan D. Wilcox
;               6 January 1985
;
0100             ORG      100H
00A0 =           CLOCK   EQU      0A0H      ; Clock base address
0000 =           FALSE   EQU      0
00FF =           TRUE    EQU      OFFH
0005 =           BDOS    EQU      5H        ; BDOS entry point
0001 =           CONSIN   EQU      1        ; Console input
0002 =           CONSOUT  EQU      2        ; Console output
0009 =           PRINTST  EQU      9        ; Print string function
000B =           GETSTAT  EQU      11       ; Console status
000A =           LF       EQU      0AH      ; Line Feed
000D =           CR       EQU      0DH      ; Carriage Return
001B =           ESC      EQU      1BH      ; Escape
;
; ***** MAIN PROGRAM *****
; *
; *
0100 113202      START:  LXI      D,GREET
0103 CD1302      CALL     PRINTIT
0106 116502      LXI      D,DATESET        ; Set the date?
0109 CD1302      CALL     PRINTIT
010C CD1B02      CALL     GETANS          ; Get answer
010F FEFF        CPI      TRUE
0111 CC2901      CZ       SETDATE
0114 117F02      LXI      D,TIMESSET      ; Set the time?
0117 CD1302      CALL     PRINTIT
011A CD1B02      CALL     GETANS          ; Get answer
011D FEFF        CPI      TRUE
011F CC6401      CZ       SETTIME
0122 119F03      LXI      D,CRLF
0125 CD1302      CALL     PRINTIT
0128 C9          DONE:   RET              ; Return to CP/M
; *
; *
; ***** SUBR SETDATE *****
;
; Set the current date
;
0129 119902      SETDATE: LXI      D,DAY    ; What day of week?
012C CD1302      CALL     PRINTIT
012F CDEB01      CALL     GETBCD          ; Get it and store
0132 D3A5        OUT      CLOCK+5        ; in clk counter
0134 11C402      LXI      D,DATE          ; What date is it?
0137 CD1302      CALL     PRINTIT
013A CDEB01      CALL     GETBCD          ; Get it and store
013D D3A6        OUT      CLOCK+6        ; in clk counter
013F 11E002      LXI      D,MONTH         ; What month is it?
0142 CD1302      CALL     PRINTIT
0145 CDEB01      CALL     GETBCD          ; Get it and store
0148 D3A7        OUT      CLOCK+7        ; in clk counter
014A 11FC02      LXI      D,YEAR          ; And the year?
014D CD1302      CALL     PRINTIT
```

(Continued on page 70)

69

Listing One

```
0150 CDDBO1      CALL    GETBCD      ; Get it and store
0153 D3A9        OUT      CLOCK+9    ; in clock RAM

0155 118A03      LXI      D,AGAIN    ; Do this over again?
0158 CD1302      CALL    PRINTIT
015B CD1B02      CALL    GETANS      ; Get answer
015E FEFF        CPI      TRUE
0160 CA2901      JZ       SETDATE    ; Yes, do it.

0163 C9          RET

;
;
; ***** SUBR SETTIME *****
;
; Set the current time
;
0164 CD9C01      SETTIME:CALL    DISPLAY    ; Show the time at start

0167 113503      LXI      D,SETHR    ; Set-hrs message
016A CD1302      CALL    PRINTIT
016D CDDBO1      CALL    GETBCD      ; Get 2 ASCII, convert to BCD
0170 D3A4        OUT      CLOCK+4    ; Put hours into clock cnter

0172 115203      LXI      D,SETMIN    ; Set-min message
0175 CD1302      CALL    PRINTIT
0178 D3B5        OUT      CLOCK+21   ; Reset seconds
017A CDDBO1      CALL    GETBCD      ; Get 2 ASCII, convert to BCD
017D D3A3        OUT      CLOCK+3    ; Put mins into clock counter

017F 116E03      LXI      D,SETSEC    ; Set-seconds message
0182 CD1302      CALL    PRINTIT
0185 CDOB02      CALL    INCHAR      ; Get key-press
0188 D3B5        OUT      CLOCK+21   ; "GO" command zero seconds

018A CD9C01      CALL    DISPLAY    ; Show the results

018D 118A03      LXI      D,AGAIN    ; Ask to do again
0190 CD1302      CALL    PRINTIT
0193 CD1B02      CALL    GETANS
0196 FEFF        CPI      TRUE
0198 CA6401      JZ       SETTIME    ; Yes, do it again

019B C9          RET

;
;
; ***** SUBR DISPLAY *****
;
; Displays current time
;
019C 111803      DISPLAY:LXI      D,TMSG    ; Time is ...
019F CD1302      CALL    PRINTIT

01A2 DBA4        IN        CLOCK+4    ; Get hours from clock
01A4 CDBC01      CALL    PRHEX      ; Print hours
01A7 3E3A        MVI      A,':'
01A9 CDO402      CALL    PCHAR

01AC DBA3        IN        CLOCK+3    ; Get minutes from clock
01AE CDBC01      CALL    PRHEX      ; Print minutes
01B1 3E3A        MVI      A,':'
01B3 CDO402      CALL    PCHAR

01B6 DBA2        IN        CLOCK+2    ; Get seconds
01B8 CDBC01      CALL    PRHEX      ; Print seconds

01BB C9          RET

;
;
; ***** SUBR PRHEX *****
;
; Converts binary value into two ASCII-hex characters
; and prints on console
```

(Continued on page 72)

Add EDITING to your Software with CSE Run-Time™

Your program can include all or a portion of the *C Screen Editor* (CSE).

CSE includes all of the basics of full screen editing plus source in C for only \$75. For only \$100 more get CSE Run-Time to cover the first 50 copies that you distribute.

Use capabilities like Full cursor control, block move, insert, search/replace or others. Portability is high for OSes, terminals, and source code.

Call for the "CSE Technical Description" and for licensing terms and restrictions.

Full Refund if
not satisfied in
first 30 days.

Call 800-821-2492

**Solution
Systems**

335-D Washington Street
Norwell, MA 02061
617-659-1571

Circle no. 75 on reader service card.

FASTER C

TRY FASTER C RISK FREE
FOR 30 DAYS WITH
OUR MONEY BACK
GUARANTEE

helps Develop and Test Lattice C Programs

Reliably Cuts Both —

- Compile times (by 15% to 55%)
- Testing time (by 12% to 37%)

"Automatic" support for new libraries by reading the .OBJ files makes support for new libraries quick and simple.

FASTER C keeps the Lattice C library and any other functions you choose in memory. It manages a jump table to replace the LINKER and immediately execute your functions. You can also CALL active functions interactively to speed your program debugging. It includes many options for configuration and control.

ONLY \$95.
CALL TOLL FREE
800-821-2492

for "Technical Description" or to order.

FASTER C is a trademark of Solution Systems

AVAILABLE FOR PC-DOS, IBM-AT,
AND ANY 256K MSDOS SYSTEM.

**Solution
Systems™**

335-D Washington St., Norwell, Mass. 02061
617-659-1571

Circle no. 93 on reader service card.

C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing
and manipulating C programs. Use C HELPER's
UNIX-like utilities which include:

DIFF and **CMP** — for "intelligent" file comparisons.
XREF — cross references variables by function and line.
C Flow Chart — shows what functions call each other.
C Beautifier — make source more regular and readable.
GREP — search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

C Helper is written in portable C and includes both full source code and executable files for \$135 for MS-DOS, IBM AT CPM-80 or CPM-86. Use VISA, Master Card or COD.

Call: 800-821-2492

**Solution
Systems™**

335-D Washington Street
Norwell, MA 02061
617-659-1571

Circle no. 94 on reader service card.

PROLOG-86™

Become Familiar in One Evening

Thorough tutorials are designed to help learn the PROLOG language quickly. The interactive PROLOG-86 Interpreter gives immediate feedback. In a few hours you will begin to feel comfortable with it. In a few days you are likely to know enough to modify some of the more sophisticated sample programs.

Sample Programs are Included like:

- an EXPERT SYSTEM
- a NATURAL LANGUAGE INTERFACE
(it generates a dBASE II "DISPLAY" command)
- a GAME (it takes less than 1 page of PROLOG-86)

PROTOTYPE Ideas and Applications QUICKLY

Serious development of experimental systems and prototypes is practical with the full syntax of PROLOG-86.

1 or 2 pages of PROLOG is often equivalent to 10 or 15 pages in "C" or PASCAL. It is a different way of thinking.

Describe the FACTS and RULES without concern for what the computer will have to do. Maybe you will rewrite in another programming language when you are done.

Programming Experience is not required but a logical mind is. PROLOG-86 supports the de facto STANDARD established in "Programming in Prolog."

AVAILABILITY: PROLOG-86 runs on MSDOS, PC DOS, IBM AT or CPM-86 machines. We provide most formats. The price of PROLOG-86 is:

Only \$125

Full Refund if not
satisfied during
first 30 days.

800-821-2492

**Solution
Systems**

335-D Washington Street
Norwell, MA 02061

617-659-1571

Circle no. 95 on reader service card.

Listing One

```

01BC F5      ;
01BD OF      PRHEX:  PUSH      PSW
01BE OF      RRC
01BF OF      RRC      ; Put 4 MSB's into 4 LSB's
01C0 OF      RRC
01C1 CDC901  CALL      PRNIB
01C4 F1      POP       PSW      ; Prnt hex equiv to 4 MSB's
01C5 CDC901  CALL      PRNIB      ; Prnt hex equiv to 4 LSB's
01C8 C9      RET

;
;
; ***** SUBR PRNIB *****
;
; Prints a nibble of Reg A
;
01C9 E60F    PRNIB:  ANI      OFH      ; Mask out top 4 bits
01CB FE0A    CPI      10      ; Is it number or letter?
01CD D2D501  JNC      LETR      ; Must be a letter if jump.
01D0 C630    ADI      '0'      ; Add offset to make ASCII
01D2 C3D701  JMP      PRNT
01D5 C637    LETR:   ADI      'A' - 10 ; Add offset to make binary
                                ; into ASCII letter
01D7 CD0402  PRNT:   CALL     PCHAR    ; Send ASCII to console
01DA C9      RET

;
;
; ***** SUBR GETBCD *****
;
; Gets 2 ASCII characters, converts them to packed BCD
;
; EXIT:      A = 2 BCD characters
;
01DB CDOB02  GETBCD:  CALL     INCHAR    ; Get character
01DE FE30    CPI      '0'
01E0 DADB01  JC       GETBCD      ; Ignore char if < ASCII 0
01E3 FE3A    CPI      '9'+1
01E5 D2DB01  JNC      GETBCD      ; Ignore char if > ASCII 9
01E8 D630    SUI      '0'      ; Make ASCII into binary
01EA E60F    ANI      OFH      ; Mask out 4 MSB's
01EC 07      RLC
01ED 07      RLC
01EE 07      RLC
01EF 07      RLC      ; Put into MSB position
01F0 47      MOV      B,A      ; and save in B
01F1 CDOB02  LOBYTE:  CALL     INCHAR    ; Get 2nd character
01F4 FE30    CPI      '0'
01F6 DAF101  JC       LOBYTE
01F9 FE3A    CPI      '9'+1
01FB D2F101  JNC      LOBYTE
01FE D630    SUI      '0'      ; Make ASCII into binary.
0200 E60F    ANI      OFH
0202 80      ADD      B      ; Combine both BCD char in A.
0203 C9      RET

;
;
; ***** SUBR PCHAR *****
;
; Prints Reg A to console
;
0204 0E02    PCHAR:  MVI      C,CONSOUT
0206 5F      MOV      E,A
0207 CD0500  CALL     BDOS
020A C9      RET

;
;
; ***** SUBR INCHAR *****
;
; Reg A gets ASCII value from keyboard
;
020B C5      INCHAR:  PUSH     B      ; Have char saved in B
020C 0E01    MVI      C,CONSIN

```

(Continued on page 74)

**THE WORLD'S FASTEST
MOST POWERFUL 8080
RELOCATING MACRO
ASSEMBLER**

SLRMAC™
ONLY \$49.95

**AT THIS PRICE, SHOULD YOU BE WASTING
YOUR TIME USING SOMETHING ELSE?**

This is what they said about Z80ASM, our Z80 assembler. Now the same features and performance are available in our Intel Mnemonic product. SLRMAC is compatible with M80 in .8080 mode, with many extensions. Too many features to list here.

To order or to find out more about our complete family of development tools, call or write:

SLR Systems

1622 N. Main St., Butler, PA 16001
(800) 833-3061, (412) 282-0864
Telex 559215 SLR SYS



C.O.D., Check or
Money Order Accepted.

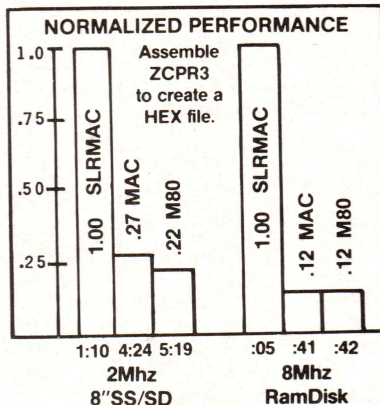
Add \$3 shipping.
Z80 CP/M compatibility required.

"... a breath of fresh air ..."

Computer Language, Feb. 85

"... in two words, I'd say speed & flexibility"

*Edward Joyce, Nov. 84
Microcomputing*



Circle no. 78 on reader service card.

U S Software®

**Software Development Products
for
Design Engineers**

8085

6809

**Embedded
Application Support**

68000

Z-80

8051

Software

8086

8096

FPAC/DPAC™
Floating Point Libraries

- IEEE Single & Double Precision Format
- Source Assembly
- Complete Math Functions
- Data Conversion Procedures

USX™
Multi-Tasking Executives

- Real-Time and ROMable
- Dynamic Task Management
- Priority Task Scheduling
- Memory Management

United States Software Corporation
5470 N.W. Innisbrook Place • Portland, Oregon 97229 • (503) 645-5043

Circle no. 91 on reader service card.

The Best Source Debugger for C

Interactive testing: enter any C expression, statement, or function call, and it is immediately executed and the result displayed. **Direct execution** allows fast and thorough testing, makes learning C a snap.

Run-time checking: execution stops upon exception, and source code displayed. Exceptions include array reference bounds, stack overflow arithmetic or floating point error, etc. Pointers are checked for null or out of range values.

Breakpoints: any number of breakpoints can be set *anywhere* in your program; breakpoints are set with screen editor, not by line numbers. Breakpoints may be conditional. Single-step by statement. Interrupt execution from keyboard. Breakpoint, exception, or interruption is always shown with source code. Examine and modify data, look at stack history. Even change your program and then resume execution!

Lint-like Compile-time checking: argument number and sizes are checked for consistency. Never mismatch source and object code.

The best feature of all: the *fastest* C interpreter is right there when you're debugging. Make changes in seconds with the integrated screen editor. Test the changes immediately, running your program at compiled speed. Save source code for your favorite compiler, or make stand-alone executable programs. Nothing else is needed. **Instant-C** is the fastest way to get working, fully debugged C programs available today.

"We sincerely feel that **Instant-C** can have a major positive impact on programmer productivity." *Computer Language*, Feb. 85, pp. 82-83.

Instant-C is only \$495. Money back for any reason in first 31 days.

**Rational
Systems, Inc.**

(617) 653-6194
P.O. Box 480
Natick, MA 01760

Circle no. 79 on reader service card.

Real-Time Clock (Listing Continued, text begins on page 56)

Listing One

```

020E CD0500      CALL    BDOS
0211 C1          POP     B
0212 C9          RET

;
;
; ***** SUBR PRINTIT *****
;
; ENTRY: DE = Start address of string
;
0213 F5 PRINTIT: PUSH    PSW
0214 0E09        MVI     C,PRINTST
0216 CD0500      CALL    BDOS          ; Print string to console
0219 F1          POP     PSW
021A C9          RET

;
; ***** SUBR GETANS *****
;
; Ask question, then call this subroutine for answer.
; Assume default answer is NO.
;
; EXIT:  A = True = FF if YES
;        A = False= 0  if NO
;
021B 0E01 GETANS: MVI     C,CONSIN
021D CD0500      CALL    BDOS          ; Get console input
0220 FE59        CPI     'Y'
0222 CA2F02      JZ      ENDTRU
0225 FE79        CPI     'y'
0227 CA2F02      JZ      ENDTRU
022A 3E00        MVI     A,FALSE
022C C33102      JMP     GOTANS        ; Not Y or y. Assume NO.
022F 3EFF        ENDTRU: MVI     A,TRUE
0231 C9          GOTANS: RET

;
; ***** CONSOLE MESSAGES *****
;
0232 ODOA50726F GREET:  DB      CR,LF,'Program to set DATE & TIME of '
0252 7265616C2D DB      'real-time clock.',LF,CR,'$'

0265 ODOA0A5365 DATESET: DB      CR,LF,LF,'Set the date? <N> --> $'
027F ODOA0A5365 TIMESET: DB      CR,LF,LF,'Set the time? <N> --> $'

0299 ODOA0A5375 DAY:    DB      CR,LF,LF,'Sunday = 01.'
02A8 ODOA456E74        DB      CR,LF,'Enter day 01 to 07. --> $'
02C4 ODOA456E74 DATE:   DB      CR,LF,'Enter date 01 to 31 --> $'
02E0 ODOA456E74 MONTH:  DB      CR,LF,'Enter month 01 to 12 --> $'
02FC ODOA456E74 YEAR:   DB      CR,LF,'Enter year 84 to 99 --> $'

0318 ODOA0A4872 TMSG:   DB      CR,LF,LF,'Hrs : Min : Sec now --> $'
0335 ODOA0A496E SETHR:  DB      CR,LF,LF,'Input HRs 00 to 23 --> $'
0352 ODOA496E70 SETMIN: DB      CR,LF,'Input Min 00 to 59 --> $'
036E ODOA507265 SETSEC: DB      CR,LF,'Press <cr> to zero sec...$'

038A ODOA0A446F AGAIN:  DB      CR,LF,LF,'Do over? <N> ... $'

039F ODOA24      CRLF:   DB      CR,LF,'$'

03A2                                END

```

End Listing One

Listing Two

```

; TIME          Display time and date of real-time clock
;
; Program by    Alan D. Wilcox
;               18 December 1984
;
; Invoke by     A>TIME          Output to Console
;               A>TIME <text>   Output to Console & LST:

```



```

                                (List time & text)
0100          ;          ORG      100H

00A0 =        CLOCK EQU      0A0H    ; Clock base address

0005 =        BDOS  EQU      5H      ; BDOS entry point
0080 =        BUFF  EQU      80H      ; Buffer

0005 =        LISTOUT EQU      5      ; List output
0009 =        PRINTST EQU      9      ; Print string function

000A =        LF     EQU      0AH      ; Line Feed
000D =        CR     EQU      0DH      ; Carriage Return
001B =        ESC    EQU      1BH      ; Escape
;
;
; ***** MAIN PROGRAM *****
; *
; *
;       Read current time and save in memory

0100 DBA4      GETTIME: IN        CLOCK+4      ; Get hours from clock cntr
0102 11D701    LXI      D,HOURS
0105 CD6D01    CALL     BCDTOASC      ; Convert to ASCII & save

0108 DBA3      IN        CLOCK+3      ; Get minutes from clk cntr
010A 11DA01    LXI      D,MINUTES
010D CD6D01    CALL     BCDTOASC      ; Convert

0110 DBA2      IN        CLOCK+2      ; Get seconds
0112 11DD01    LXI      D,SECONDS
0115 CD6D01    CALL     BCDTOASC

;       Read day of week and save literal equiv. in memory

0118 DBA5      GETDAY: IN        CLOCK+5
011A 11E201    LXI      D,DAY          ; Where to save the result
011D 211302    LXI      H,LITDAY      ; Location of literal weekday
0120 CD7F01    CALL     NTOLIT        ; Convert number to literal

;       Read date and save in memory

0123 DBA6      GETDATE: IN        CLOCK+6
0125 11EE01    LXI      D,DATE        ; Where to save result
0128 CD6D01    CALL     BCDTOASC      ; Convert to ASCII and save

;       Read month and save literal equivalent in memory

012B DBA7      GETMON: IN        CLOCK+7
012D 11F101    LXI      D,MONTH        ; Where to save result
0130 215902    LXI      H,LITMON      ; Location of literal months
0133 CD7F01    CALL     NTOLIT        ; Convert to literal

;       Read year from clock RAM & save in memory

0136 DBA9      GETYR: IN        CLOCK+9
0138 11FD01    LXI      D,YEAR        ; Where to save result
013B CD6D01    CALL     BCDTOASC      ; Convert to ASCII and save

;       Send time message to console

013E 11CE01    CONSL: LXI      D,MSGL      ; DE gets adr of message
0141 0E09      MVI      C,PRINTST
0143 CD0500    CALL     BDOS          ; Print string to console

;       Send time msg to printer if cmd string required it

0146 3A8000    PRINT: LDA      BUFF      ; Was cmd 'TIME <text>'
0149 FE00      CPI      0              ; Done if nothing there
014B CA6C01    JZ       DONE

014E 4F        MOV      C,A            ; Put buffer count in C
014F 218100    LXI      H,BUFF+1      ; HL is source of text
0152 110102    LXI      D,TEXT        ; DE is destin for text

0155 7E        MOVETXT: MOV     A,M      ; Get byte
0156 12        STAX     D              ; Save it

```

(Continued on next page)

Listing Two

```

0157 23          INX      H
0158 13          INX      D
0159 0D          DCR      C
015A C25501      JNZ      MOVETXT      ; Do until C is 0

015D 3E24        MVI      A,'$'      ; End-text marker
015F 12          STAX     D

0160 21BD01      LXI      H,MSGR      ; Go ahead to printer
0163 CDA701      CALL     LISTIT
0166 21BA01      LXI      H,CRLF      ; Finish the line
0169 CDA701      CALL     LISTIT

016C C9          DONE:   RET          ; Return to CP/M
; *
; *

; ***** SUBR BCDTOASC *****
;
; Converts binary value into two ASCII-hex characters
;
; ENTRY:  Reg A = 2 packed-BCD characters
;         DE = Adr of where to store ASCII answers
;
; EXIT:   (DE) = MS char
;         (DE+1) = LS char
;
BCDTOASC:
016D F5          PUSH     PSW
016E 0F          RRC
016F 0F          RRC      ; Put 4 MSB's into 4 LSB's
0170 0F          RRC
0171 0F          RRC
0172 E60F        ANI      0FH      ; Mask out top 4 bits
0174 C630        ADI      '0'      ; Add offset to make ASCII
0176 12          STAX     D      ; Save the result
0177 F1          POP      PSW      ; Get the next character
0178 E60F        ANI      0FH
017A C630        ADI      '0'      ; Make ASCII
017C 13          INX      D
017D 12          STAX     D      ; Save it in next memory loc.
017E C9          RET

; ***** SUBR NTOLIT *****
;
; Converts packed-BCD number (1 to 12) to its equiv
; literal string and saves in specified memory.
;
; ENTRY:  A = 2 packed-BCD numbers from clock
;         DE = Adr of where to store literal results
;         HL = Pntr to 10-char literal strings
;
017F 47          NTOLIT: MOV     B,A      ; Save data
0180 E610        ANI      00010000B      ; See if have tens digit
0182 CA8B01      JZ       SMALLNR      ; Jump if not
0185 78          MOV      A,B
0186 C60A        ADI      10      ; Add 10 to low nibble so
0188 C38C01      JMP      BINARY      ; it's binary now.
018B 78          SMALLNR:MOV     A,B

018C E60F        BINARY: ANI      0FH      ; Mask off top nibble
018E D601        SUI      1      ; Make number 0 to 11 max
0190 CA9B01      JZ       MOVE      ; At first already

0193 010A00      LXI      B,10      ; 10 literals in word
0196 09          NEXT:   DAD      B      ; Add it to HL for next word
0197 3D          DCR      A      ; until get to proper one.
0198 C29601      JNZ      NEXT

```

(Continued on page 78)

NOW
for Microsoft
C version 3.0

the source debugger for lattice C

Your time and convenience come first! The MSD C Debugger™ is the last, and perhaps final, word in programming assistance for Lattice C users. C Debugger produces a high level view of C programs via function names, line numbers, variable names and C data types, plus a low-level view of machine addresses and instructions for testing assembler language functions.

More features include:

- All documentation is prepared for programmers.
- Online help screen throughout the process.
- Capability to single step through your program.
- Set break points, examine registers and variables.

\$165.00 + \$3.50 shipping



To order, call or write:

MICRO-SOFTWARE DEVELOPERS, INC.
214½ W. Main St. • St. Charles, IL 60174
312/377-5151

Lattice C is a trademark of Lattice, Inc.
Microsoft is a trademark of Microsoft Corp.

Circle no. 134 on reader service card.

Dr. Dobb's Journal Subscription Problems?

No Problem!



Give us a call and we'll straighten it out. Today.

Outside California
CALL TOLL FREE: 800-321-3333

Inside California
CALL: 619-485-9623 or 566-6947

Write it once!

MasterFORTH

Portable programming environment



Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest. If

you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.



MasterFORTH is a state-of-the-art implementation of the Forth computer language.

Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in macro assembler to make it even faster. MasterFORTH's relocatable utilities, transient definitions, and headerless code



let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint, and trace your way through most programming problems. A string package, file interface, and full screen editor are all standard features.

CP/M

MasterFORTH exactly matches the Forth-83 Standard dialect described in *Mastering Forth* by Anderson and Tracy (Brady, 1984). The standard package includes the book and over 100 pages of supplementary documentation.

MasterFORTH standard package

Macintosh	\$125
IBM PC and PC Jr. (MS DOS 2.1)	125
Apple II, II+, IIe, IIc (DOS 3.3)	100
CP/M 2.X (in several formats)	100
Commodore 64	100

Extensions

Floating Point (1984 FVG standard)	\$40
Graphics (Apple II series)	40
Module relocater (with utility sources)	60
Printed source listing (each)	35

Publications

<i>Mastering Forth</i> (additional copies)	\$18
<i>Thinking Forth</i> by Leo Brodie	16
<i>Forth-83 International Standard</i>	15
<i>Rochester Bibliography</i> , 2nd ed.	15
<i>1984 Rochester Conference</i>	25
<i>1984 J1 of Forth Appl. & Res.</i> 2(2)	15
<i>1983 FORML Conference</i>	25



MICROMOTION

12077 Wilshire Blvd., #506
Los Angeles, CA 90025
(213) 821-4340

Circle no. 53 on reader service card.

Listing Two

```

019B 010A00      MOVE:   LXI       B,10           ; Move 10 letters only
019E 7E          MOVDAT: MOV      A,M           ; HL pts to desired word
019F 12          STAX     D                   ; DE pts to destination mem.
01A0 13          INX      D
01A1 23          INX      H
01A2 0D          DCR      C                   ; Proper literal word now
01A3 C29E01      JNZ      MOVDAT              ; moved to memory to print
01A6 C9          RET

```

```

; ***** SUBR LISTIT *****
;
;       Lists to printer until '$' encountered.
;
;       ENTRY:  HL = string address
;
01A7 7E          LISTIT: MOV      A,M           ; Get the letter to be sent
01A8 FE24        CPI       '$'                ; End of text yet?
01AA CAB901      JZ        LISTEND
01AD E5          PUSH     H                   ; Must retain HL!
01AE 5F          MOV      E,A
01AF 0E05        MVI      C,LISTOUT
01B1 CD0500      CALL     BDOS                ; Print the char
01B4 E1          POP      H
01B5 23          INX      H
01B6 C3A701      JMP      LISTIT              ; Keep going until '$'
01B9 C9          LISTEND:RET

```

```

; ***** MESSAGE AREA *****
;
01BA 0D0A24      CRLF:     DB          CR,LF,'$'
01BD 20202020    MSGR:     DB          ' '
01CC 2020        DB          ' '
01CE 20202020    MSGL:     DB          ' '
01D7             HOURS:    DS          2
01D9 3A          DB          ': '
01DA             MINUTES:  DS          2
01DC 3A          DB          ': '
01DD             SECONDS:  DS          2
01DF 202020      DB          ' '
01E2             DAY:      DS          10
01EC 2020        DB          ' '
01EE             DATE:     DS          2
01F0 20          DB          ' '
01F1             MONTH:    DS          10
01FB 3139        DB          '19'
01FD             YEAR:     DS          2
01FF 2020        DB          ' '
0201 20202020    TEXT:     DB          ' '
0210 0D0A24      DB          CR,LF,'$'

```

```

; ***** LITERALS *****
;
;       Pack with NULLs for even spacing on print line
;
0213 53756E6461 LITDAY:    DB          'Sunday',' ',0,0,0
021D 4D6F6E6461 DB          'Monday',' ',0,0,0
0227 5475657364 DB          'Tuesday',' ',0,0
0231 5765646E65 DB          'Wednesday',' '
023B 5468757273 DB          'Thursday',' ',0
0245 4672696461 DB          'Friday',' ',0,0,0
024F 5361747572 DB          'Saturday',' ',0
0259 4A616E7561 LITMON:    DB          'January ',0,0
0263 4665627275 DB          'February ',0
026D 4D61726368 DB          'March ',0,0,0,0

```


0277	417072696C	DB	'April ',	0,0,0,0
0281	4D61792000	DB	'May ',	0,0,0,0,0,0
028B	4A756E6520	DB	'June ',	0,0,0,0,0
0295	4A756C7920	DB	'July ',	0,0,0,0,0
029F	4175677573	DB	'August ',	0,0,0
02A9	5365707465	DB	'September '	
02B3	4F63746F62	DB	'October ',	0,0
02BD	4E6F76656D	DB	'November ',	0
02C7	446563656D	DB	'December ',	0

02D1

END

End Listing Two

Listing Three

```

; TIMELIST      Program to read a file and send it out
;               to printer with time/date and program name
;
; Program by    Alan D. Wilcox
;               29 October 1984
;
; Usage         A> TIMELIST filespec.txt
;

0100            ORG      100H

00A0 =          CLOCK   EQU      0A0H      ; Clock base address

0000 =          FALSE   EQU      0
FFFF =          TRUE    EQU      NOT FALSE

FFFF =          TOLPTR  EQU      TRUE      ; Make FALSE to send all output to
;               console rather than to line prntr

0000 =          KEEPFF  EQU      FALSE     ; Make TRUE to use form feeds in
;               orig text. TRUE also cancels line
;               counting by this program.

0005 =          BDOS    EQU      0005H     ; BDOS entry point

0002 =          CONSOUT EQU      2         ; Console Output
0005 =          LISTOUT EQU      5         ; List Output
0009 =          PRINTST EQU      9         ; Print String Function
000B =          GETSTAT EQU      11        ; Console Status
000F =          OPENF   EQU      15        ; Open File
0014 =          READF   EQU      20        ; Read Next Record

005C =          FCB     EQU      5CH       ; File Control Block Address
007C =          FCBCR   EQU      FCB+32    ; Current (next) record for r/w
0080 =          BUFF    EQU      80H       ; Input Disk Buffer Address

000C =          TSIZE   EQU      12        ; Title block size
0003 =          SKIP    EQU      3         ; Added number of spcs at left margin
0050 =          COLS    EQU      80        ; Max possible columns per page
003C =          LPAGE   EQU      60        ; Lines to print per page
0008 =          NRSPCS  EQU      8         ; Convert tab to NRSPCS spaces

0009 =          TAB     EQU      09H       ; Horizontal Tab
000A =          LF      EQU      0AH       ; Line Feed
000B =          VT      EQU      0BH       ; Vertical Tab (alt FF)
000C =          FF      EQU      0CH       ; Form Feed
000D =          CR      EQU      0DH       ; Carriage Return
001A =          CNTLZ   EQU      1AH       ; End of File (cntl-z)
001B =          ESC     EQU      1BH       ; Escape
0020 =          SPC     EQU      20H       ; Space
;
;
; ***** MAIN PROGRAM *****
; *
; *

0100 210000     STACK:  LXI      H,0        ; Stack in safe place while
0103 39         DAD      SP              ; using disk buffer space.
0104 227904     SHLD     OLDSTK         ; Save old stack position

```

(Continued on next page)

Listing Three

```
0107 319F04          LXI      SP,NEWSTK          ; Use new stack pointer
010A CD1B01          CALL     OPENFILE           ; Open program to be read
010D CD3B01          CALL     SAVETITLE          ; Save pgm name as the title
0110 CD7901          CALL     FIRSTLINE         ; Prnt time/date/title line
0113 CDD301          CALL     PRINTLOOP         ; Read & print file
0116 2A7904  QUIT:    LHL      OLDSTK           ; Get the old stack pntr in
0119 F9              SPHL                     ; HL & put in on the stack.
011A C9              RET                        ; Return to CP/M

; *
; *
;
; ***** SUBR OPENFILE *****
;
OPENFILE:            ; Open the file requested

011B AF              XRA      A                  ; Zero accumulator
011C 327C00          STA      FCBCR             ; Zero file record count
011F 115C00          LXI      D,FCB
0122 OEOF            MVI      C,OPENF          ; Open the file
0124 CD0500          CALL     BDOS              ; Get OFFH in Accum if error
0127 FEFF            CPI      OFFH
0129 C23501          JNZ      OKOPEN

012C 113D03          LXI      D,OPENERR         ; Had error in file opening
012F CD3503          CALL     PRINTIT           ; Print error message
0132 CD1601          CALL     QUIT              ; Restore stack and get out

0135 3E80            OKOPEN: MVI     A,BUFF      ; Set position of input
0137 327B04          STA      INBUFFT          ; buffer pointer
013A C9              RET

;
;
; ***** SUBR SAVETITLE *****
;
SAVETITLE:           ; Save name of program as title

013B OEOC            MVI      C,TSIZE          ; Clear TSIZE space in memory
013D 21AA03          LXI      H,HEADER         ; starting at HEADER.
0140 3620            CLRBLK: MVI     M,SPC      ; Fill with spaces
0142 23              INX      H
0143 OD              DCR      C
0144 C24001          JNZ      CLRBLK
0147 3A8300          LDA      BUFF+3           ; See if drive in cmd line
014A FE3A            CPI      ':'              ; Z set if true
014C C25A01          JNZ      NODRIV
014F 118400          LXI      D,BUFF+4         ; DE pts to pgm name only
0152 3A8000          LDA      BUFF            ; Get count of chars
0155 D603            SUI      3                ; Sub cnt, space, drive spec
0157 C36201          JMP      ADJSIZE

015A 118200          NODRIV: LXI      D,BUFF+2   ; DE points to pgm name
015D 3A8000          LDA      BUFF            ; Get count of chars
0160 D601            SUI      1                ; Subtr count & space

0162 FEOC            ADJSIZE: CPI      TSIZE    ; Adjust char cnt so it fits
0164 F26B01          JP        BIG             ; Jump if count > TSIZE
0167 4F              MOV      C,A             ; Acceptable count into C
0168 C36D01          JMP      OKSIZE
016B OEOC            BIG:    MVI      C,TSIZE   ; Title is max size

016D 21AA03          OKSIZE: LXI      H,HEADER   ; Destination for title move

0170 1A              MTITLE: LDAX     D         ; Get the character
0171 77              MOV      M,A             ; and save it as header
0172 23              INX      H
0173 13              INX      D
```



```

0174 OD          DCR      C
0175 C27001      JNZ      MTITLE      ; Title saved when zero
0178 C9          RET

;
; ***** SUBR FIRSTLINE *****
;
FIRSTLINE:      ; Read current time and save in memory

0179 DBA4          IN      CLOCK+4      ; Get hours from clock
017B 116B03        LXI      D,HOURS
017E CDE102        CALL     BCDTOASC     ; Convert to ASCII & save

0181 DBA3          IN      CLOCK+3      ; Get minutes from clock
0183 116E03        LXI      D,MINUTES
0186 CDE102        CALL     BCDTOASC     ; Convert

0189 DBA2          IN      CLOCK+2      ; Get seconds
018B 117103        LXI      D,SECONDS
018E CDE102        CALL     BCDTOASC

;      Read day of week and save literal equiv. in memory

0191 DBA5          IN      CLOCK+5
0193 118003        LXI      D,DAY      ; Where to save the result
0196 21BB03        LXI      H,LITDAY   ; Location of literal weekday
0199 CDF302        CALL     NTOLIT     ; Convert number to literal

;      Read date and save in memory

019C DBA6          IN      CLOCK+6
019E 118B03        LXI      D,DATE      ; Where to save result
01A1 CDE102        CALL     BCDTOASC     ; Convert to ASCII and save

;      Read month and save literal equivalent in memory

01A4 DBA7          IN      CLOCK+7
01A6 118E03        LXI      D,MONTH     ; Where to save result
01A9 210104        LXI      H,LITMON   ; Location of literal months
01AC CDF302        CALL     NTOLIT     ; Convert to literal

;      Read year from clock RAM & save in memory

01AF DBA9          IN      CLOCK+9
01B1 119A03        LXI      D,YEAR
01B4 CDE102        CALL     BCDTOASC     ; Convert to ASCII and save

;      Clear spaces at left margin before printing time

01B7 0E03          MVI      C,SKIP
01B9 216803        LXI      H,TOPLINE
01BC 3620          CLRSKP: MVI      M,SPC      ; Fill with spaces
01BE 23            INX      H
01BF 0D            DCR      C
01C0 C2BC01        JNZ      CLRSKP

IF TOLPTR          ; Send time message to line printer
01C3 216803        LXI      H,TOPLINE      ; HL gets adr of message
01C6 CD1B03        CALL     LISTIT      ; To printer
ENDIF
;
IF NOT TOLPTR      ; Send time message to console
01C9 216803        LXI      D,TOPLINE      ; DE gets adr of message
01CB CD1B03        CALL     PRINTIT      ; Print to console
ENDIF
;      Set line cntr to 3 to account for title block lines

01C9 3E03          MVI      A,3
01CB 327E04        STA      LINECNT

;      Initialize column counter

01CE AF            XRA      A      ; Zero accum
01CF 327D04        STA      COLCNT      ; Column counter zero
01D2 C9            RET
;
;

```

(Continued on next page)

Real-Time Clock (Listing Continued, text begins on page 56)

Listing Three

```

; ***** SUBR PRINTLOOP *****
;
PRINTLOOP:      ; Will stay in this loop until either break
                  ; from console or file is completely printed

;      Check console for break. Exit without losing stack

01D3 OE0B      MVI      C,GETSTAT
01D5 CD0500     CALL     BDOS      ; Accum LSB=1 if brk request
01D8 0F        RRC      ; Put LSB into carry
01D9 DA5702     JC       DONE      ; Leave program

;      Check to see if need spaces at start of left margin

01DC 3A7D04     LDA      COLCNT    ; Current column count
01DF FE03      CPI      SKIP      ; Sign set while COLCNT< SKIP
01E1 F2ED01     JP       CKLINE    ; Jump when COLCNT>=SKIP
01E4 3C        INR      A          ; Increment col counter
01E5 327D04     STA      COLCNT
01E8 3E20      MVI      A,SPC
01EA C35102     JMP      GPRNT      ; Go print space & continue

;      Check for proper lines per page. Send FF if needed

01ED 3A7E04     CKLINE: LDA     LINECNT ; Current line count
01F0 FE3C      CPI      LPAGE      ; Max lines per page
01F2 C20102     JNZ      DATA      ; Get data if not equal yet

IF KEEPFF
    JMP      DATA      ; Don't bother counting.
ENDIF          ; Use FF's in source text.

01F5 AF        XRA      A          ; Zero accum
01F6 327E04     STA      LINECNT    ; Reset line cntr when equal
01F9 327D04     STA      COLCNT     ; Reset column counter too
01FC 3E0C      MVI      A,FF
01FE C35102     JMP      GPRNT      ; Print FF & continue

;      Get the data and filter out various characters

0201 CD5B02     DATA: CALL     GETNB ; Get the next byte.
0204 DA5702     JC       DONE      ; Carry set if EOF
0207 FE0D      CPI      CR
0209 CA5102     JZ       GPRNT      ; Print the CR

020C FE0A      CKLF:  CPI      LF
020E C22102     JNZ      CKFF      ; Jump if not a LF
0211 3A7E04     LDA      LINECNT
0214 3C        INR      A          ; Increment line counter
0215 327E04     STA      LINECNT
0218 AF        XRA      A
0219 327D04     STA      COLCNT     ; Set col counter to zero
021C 3E0A      MVI      A,LF      ; Restore data
021E C35102     JMP      GPRNT      ; Print the LF

IF KEEPFF
    CKFF:  JMP      LNWID      ; Go and print FF & VT chars
ENDIF

;
IF NOT KEEPFF
    CKFF:  CPI      FF
0221 FE0C      JZ       DATA      ; If FF, ignore. Get new data
0223 CA0102     CPI      VT
0226 FE0B      JZ       DATA      ; If VT, ignore it too.
0228 CA0102     ENDIF

;      Check for proper width of line

022B F5        LNWID: PUSH     PSW
022C 3A7D04     LDA      COLCNT    ; Get current column count
022F FE50      CPI      COLS      ; Max allowed per line
0231 C24202     JNZ      NOTMAX

```

(Continued on page 84)

GRAPHICS FROM YOUR
DOT MATRIX PRINTER

HPlot

A PLOTTER EMULATION PROGRAM
FOR YOUR OKIDATA, PROWRITER,
GEMINI, OR EPSON PRINTER.

- * POWERFUL HP-GL PLOTTER SYNTAX:
SCALING, LINETYPES, WINDOWS,
ETC; LABELS ANY SIZE, SLANT,
OR ROTATED.
- * FAST! GRAPHS IN FOUR MINUTES.
- * HI-RES MODE: UP TO 136x144 DPI.
- * PLOT SIZES 11"x14" TO 7"x48".
- * 80+ PAGE ILLUSTRATED MANUAL.
- * SOURCE CODE IN C FOR
PROGRAMS THAT USE
HPlot TO MAKE PIE CHARTS, GRAPHS, ETC.
- * REQUIRES 54K Z80 CP/M 2.2.
OTHER PRINTERS AND OS'S SOON!
- * AVAILABLE IN 8" SSD AND MOST
5.25" 48 TPI FORMATS.

\$49.95 PPD. OH RES ADD 5% TAX

ORDINATE SOLUTIONS
MAIN P.O. BOX 0308, OBERLIN, OH 44074

THIS AD WAS PREPARED ON AN OKIDATA 92.

TURBO ASSEMBLER

Introducing **FAST ASSEM-86™**, the **TURBO PASCAL™** of IBM PC assemblers. **FAST ASSEM-86™ (FASM)** is significantly faster and easier to use than the IBM Macro-Assembler (MASM). Whether you are new to assembly language and want to quickly write a small assembly language routine, or are an experienced MASM user tired of waiting months to assemble large files, **FAST ASSEM-86** will bring the excitement back to assembly language.

FAST ASSEM-86 IS MUCH FASTER:

- How fast is **FASM™**? The graph below shows relative assembly times for a 48K source file. For large files like this we blow MASM's doors off at 3 times their speed. For smaller 8K files we positively vaporize them at 6 times their speed.

FASM™	(110 sec.)	
MASM	(340 sec.)	

- **FAST ASSEM-86** is faster for the following reasons: (1) Written entirely in assembly language (unlike MASM). (2) Editor, assembler and source file always in memory so you can go instantly from editing to assembling and back. (3) Eliminates the time needed to LINK programs. Executable .COM files can be created directly. (Also creates .OBJ files compatible with the IBM linker).

FAST ASSEM-86 IS EASIER TO USE:

FASM includes many other features to make your programming simpler.

- Listings are sent directly to screen or printer. Assemblies can be single stepped and examined without having to leave the editor.
- Access the built in cross reference utility from the editor.
- Full support of 186 and 286 (real mode) instructions.
- Both Microsoft and 8087 floating point formats are supported. 8087 and 287 instructions supported directly without macros for faster assembly.
- Calculator mode: Do math in any radix even using symbols from the symbol table.
- Direct to memory assembly feature lets you test execute your code from editor.
- Coming soon: A coordinated symbolic debugger.

COMPATIBILITY: **FASM** is source code compatible with MASM and supports macros, records and structures.

Introductory Price \$49
With .OBJ Capability \$99

Speedware™

IBM, Turbo Pascal, Microsoft trademarks of IBM Corp., Borland Intern., Microsoft Corp.

Dealer inquiries welcome
118 Buck Circle
Folsom, CA 95630
(916) 988-7426

Epsilon

The Emacs-Like Text Editor For Programmers Who Don't Like to Wait!!

State of the Art Text Editor

Epsilon is an exciting new text editor designed to make programmers more productive. Epsilon is faster than Brief, faster than Mince, faster than Gosling Emacs, and faster than the editor you're using now!

Concurrent Processes!

Epsilon lets you compile while you edit! You can run compilers, assemblers, linkers, and almost any other program that isn't screen oriented, all under Epsilon's control, while you edit your files!

With Epsilon you don't wait for programs like compilers to finish. Use Epsilon's concurrent process command, and while the compiler runs, you can continue to examine and edit files. Any errors in the compilation are displayed immediately, and Epsilon gives you the opportunity to correct them **while the compiler continues to run.** With *Epsilon*, you're finished correcting errors when other editors first let you start.

Powerful Commands

Epsilon has over 125 commands instantly available. Epsilon can manipulate words, sentences, and paragraphs easily. Epsilon will automatically save text you have deleted in a "ring" of kill-buffers, so that you can retrieve it later. It will help you avoid syntax errors by displaying matching parentheses. And best of all, *Epsilon's* macros let you define your own commands, which can be loaded automatically each time you start Epsilon.

Speed with No Limits.

Epsilon reads and writes files 25% to 600% faster than competing editors. From its convenient keyboard macros to its facility that completes the names of commands, files and buffers, to its optimized incremental search, Epsilon has been designed for programming ease and speed.

There's no limit to the number or the size of buffers you can have. Each buffer can hold a different file, or different versions of the same file. You can create as many windows as will fit on the screen, and display different buffers in each. And should you run out of memory, Epsilon will create and automatically utilize a swap file.

Speed Comparison (In Seconds) with Other Editors

	Epsilon	Brief	Mince	Emacs
Start-up	2.60	4.11	1.43	24.93
Read 21K file	1.06	1.33	8.95	7.52
Write 21K file	2.11	14.30	6.05	7.95
Next Screen	.19	.24	1.33	1.80
String Search	3.85	7.04	4.49	8.41
I-Search	3.85	--	--	8.73
First Help	8.30	12.33	--	--
Other Helps	.20	11.64	--	--

Epsilon runs on IBM PC's, XT's, AT's and compatibles with PC-DOS 2.0 or above and requires 192K of memory.

Epsilon's price is only \$195.00.
ALL MAJOR CREDIT CARDS ACCEPTED.



Lugaru Software, Ltd.

5227 Fifth Avenue, Suite 12 / P. O. Box 110037
Pittsburgh, Pa. 15232

(412) 621-5911

Listing Three

```

0234 CDC902          CALL    PRCRLF          ; If equal, do CR/LF, reset
                                           ; the col counter to zero,
                                           ; increment line counter.
0237 3A7B04          LDA     INBUFPT         ; Set buffer pointer back
023A 3D              DCR     A               ; to ignore this char.
023B 327B04          STA     INBUFPT
023E F1              POP     PSW
023F C3D301          JMP     PRINTLOOP       ; Reset stack and go back
                                           ; thru read loop

0242 F1              NOTMAX: POP    PSW

0243 FE09            CKTAB: CPI     TAB
0245 CC8602          CZ      TABTOSP         ; If TAB, expand it to spaces
0248 F5              PUSH    PSW           ; Save data byte now in accum
0249 3A7D04          LDA     COLCNT
024C 3C              INR     A             ; Increment column counter
024D 327D04          STA     COLCNT
0250 F1              POP     PSW           ; Restore data for print

0251 CD2E03          GOPRNT: CALL   PCHAR
0254 C3D301          JMP     PRINTLOOP

0257 CDC902          DONE:  CALL   PRCRLF       ; Print CR/LF to wrapup
025A C9              RET

;
;
; *****          SUBR GETNB          *****
;
; Gets the next byte from memory or EOF flag if done
;
; EXIT:  A = Next file byte fetched from memory
;        F = Carry set if End-of-File

025B 3A7B04          GETNB: LDA     INBUFPT     ; Get current buff rel ptr
025E FE80            CPI     BUFF             ; Is it to end of rec yet?
0260 C27302          JNZ     GET              ; Get next byte if not.

; Read a 128-byte record from disk

0263 115C00          LXI     D,FCB           ; DE gets FCB adr. Reads 128
0266 0E14            MVI     C,READF         ; bytes into mem starting
0268 CD0500          CALL    BDOS            ; at BUFF adr
026B B7              ORA     A               ; A=0 if record read OK, so
026C CA7302          JZ      GET              ; go get the info.
                                           ; Else if A#0, then is EOF.
026F 37              EOF:   STC
0270 C38502          JMP     GETEND           ; Set carry to indicate EOF.

GET:                 ; Read the byte at BUFF + Reg A's relative offset

0273 5F              MOV     E,A             ; DE contains rel position of
0274 1600            MVI     D,0             ; pntr into disk buff space
0276 3C              INR     A
0277 327B04          STA     INBUFPT         ; Incr and save new pointer.
027A 218000          LXI     H,BUFF          ; Start addr of buffer space
027D 19              DAD     D               ; plus rel offset ---> HL
027E 7E              MOV     A,M            ; Get byte ptd to by HL
027F FE1A            CPI     CNTLZ          ; Check for EOF within record
0281 CA6F02          JZ      EOF             ; & exit with carry bit set

0284 B7              ORA     A               ; Reset carry bit
0285 C9              GETEND: RET

;
;
; *****          SUBR TABTOSP         *****
;
; TABTOSP:           ; Converts a tab to every NRSPCS columns

0286 3A7D04          LDA     COLCNT         ; Which column are we in?
0289 DE03            SBI     SKIP           ; Make relative offset
028B 5F              MOV     E,A             ; Put it in E
028C 1608            MVI     D,NRSPCS

```

(Continued on page 86)

TOTAL CONTROL:

FORTH: FOR Z-80®, 8086, 68000, and IBM® PC

Complies with the New 83-Standard

**GRAPHICS • GAMES • COMMUNICATIONS • ROBOTICS
DATA ACQUISITION • PROCESS CONTROL**

● **FORTH** programs are instantly portable across the four most popular microprocessors.

● **FORTH** is interactive and conversational, but 20 times faster than BASIC.

● **FORTH** programs are highly structured, modular, easy to maintain.

● **FORTH** affords direct control over all interrupts, memory locations, and i/o ports.

● **FORTH** allows full access to DOS files and functions.

● **FORTH** application programs can be compiled into turnkey COM files and distributed with no license fee.

● **FORTH** Cross Compilers are available for ROM'ed or disk based applications on most microprocessors.

Trademarks: IBM, International Business Machines Corp.; CP/M, Digital Research Inc.; PC/Forth+ and PC/GEN, Laboratory Microsystems, Inc.

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities and 200 page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

Z-80 FORTH for CP/M® 2.2 or MP/M II, \$100.00;
8080 FORTH for CP/M 2.2 or MP/M II, \$100.00;
8086 FORTH for CP/M-86 or MS-DOS, \$100.00;
PC/FORTH for PC-DOS, CP/M-86, or CCPM, \$100.00; **68000 FORTH** for CP/M-68K, \$250.00.

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly.

PC FORTH + \$250.00
8086 FORTH + for CP/M-86 or MS-DOS \$250.00
68000 FORTH + for CP/M-68K \$400.00

Extension Packages available include: software floating point, cross compilers, INTEL 8087 support, AMD 9511 support, advanced color graphics, custom character sets, symbolic debugger, telecommunications, cross reference utility, B-tree file manager. Write for brochure.



Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to (213) 306-7412



Circle no. 55 on reader service card.

\$5.00 C Compiler

Due to popular demand, **Dr. Dobb's Journal** has reprinted its most-asked-for C compiler articles by Ron Cain and J. E. Hendrix, each for only \$5.00.

Ron Cain's C compiler from sold-out 1980 issues #45 and #48 includes "A Small C Compiler for the 8080s" and "Runtime Library for the Small C Compiler."

The J. E. Hendrix reprint includes part two of "Small-C Compiler v.2" from sold out issue #75 and completes the first part of the compiler article from issue #74 which is included in Dr. Dobb's Bound Volume 7. To Order: Enclose \$5.00 for each copy with this coupon and send to:

Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303
Outside U.S., add \$2.00 per copy for shipping and handling.

Please send _____ copy(ies) of the Ron Cain Reprint, and
_____ copy(ies) of the J. E. Hendrix reprint to:

Name _____

Address _____

City _____ State _____ Zip _____

ALL REPRINT ORDERS MUST BE PREPAID.

Please allow 6-9 weeks for delivery.

105



PROGRAMMER'S UTILITIES

especially for Turbo Pascal on
IBM PC/XT/AT and compatibles

MORE POWERFUL THAN UNIX UTILITIES!!!

Whether you are a

-- Student -- Hobbyist -- Professional Software Developer --

THESE UTILITIES WILL IMPROVE YOUR
PROGRAMMING PRODUCTIVITY!!!

These Powerful, Ready-to-Use programs fully support Turbo Pascal versions 2.0 and 3.0, and MSDOS 2.X and 3.0. Here's what you get:

Pretty Printer

Standardize capitalization, indentation, and spacing of source code. Don't waste your own time! Several adjustable parameters to suit your tastes (works with any standard Pascal source).

Program Structure Analyzer

Find subtle problems the compiler doesn't: uninitialized and unused variables, modified value parameters, "sneaky" variable modification, redefined standard identifiers. Also generates a complete variable cross reference and a program hierarchy diagram. Interactive or write to file (works with any standard Pascal source).

Execution Timer

Obtain a summary of time spent in each procedure and function of your program, accurate to within 200 microseconds. Also counts number of calls to each subprogram. Fully automatic.

Execution Profiler

Obtain a graphic profile of where your program spends its time. Interactive, easy-to-use. Identify weak code at the instruction level. (Profiler and Timer for Turbo Pascal Source code only.)

Command Repeater

Go beyond MSDOS batch files to combine a powerful text parser with general-purpose command execution capability. Use to copy, print or delete across subdirectories. "make" programs and more.

Pattern Replacer

Find and REPLACE versatile regular expression patterns in any text file. Supports nesting, alternation, tagged words and more. Over a dozen programmer's applications included.

Difference Finder

Find differences between two text files, and optionally create an EDLIN script which rebuilds one from the other. Disregard white space, case, arbitrary characters and Pascal comments if desired.

Super Directory

Replace PCDOS DIR command with extended pattern matching, sort capability, hidden file display, date filtering, and more.

File Finder

Locate files anywhere in the subdirectory tree and access them with a single keystroke. Display the subdirectory tree graphically.

AVAILABLE IN SOURCE AND EXECUTABLE FORMAT

Executable: \$55 COMPLETE including tax and shipping. Compiled and ready to run, includes user manual, reference card and one 5 1/4" DSDD disk. Ideal for programmers not using Turbo.

Source: \$95 COMPLETE including tax and shipping. Includes all of the above, and two additional DSDD disks. Disks include complete Turbo Pascal source code, detailed programmer's manual (on disk) and several bonus utilities. Requires Turbo Pascal 2.0 or 3.0.

Requirements: MSDOS 2.X or 3.0, 192K RAM — programs run in less RAM with reduced capacity. Two drives or hard disk recommended.

TO ORDER:

VISA/MasterCard orders, call 7 days toll-free 1-800-538-8157 x830. In California, call 1-800-672-3470 x830 any day.

Or mail check/money order to:

TurboPower Software
478 W. Hamilton Ave., Suite 196
Campbell, CA 95008

Circle no. 81 on reader service card.

Listing Three

```

028E CDB002          CALL    MODULO          ; Do (COLCNT)mod NRSPCS
                                      ; Remainder is in Reg C
0291 3E08            MVI     A,NRSPCS
0293 91              SUB     C                ; A has total # spaces to do
0294 3D              DCR     A                ; Pickup space after RET
0295 4F              MOV     C,A              ; C has # spaces to do now
0296 FE01            CPI     1                ; Set sign if need less than
0298 FAAD02          JM      ENDTAB           ; one space printed.

029B 3E20            SPACES: MVI    A,SPC
029D C5              PUSH    B
029E CD2E03          CALL    PCHAR
02A1 C1              POP     B
02A2 3A7D04          LDA     COLCNT           ; Keep column count current
02A5 3C              INR     A                ; while sending spaces
02A6 327D04          STA     COLCNT
02A9 0D              DCR     C
02AA C29B02          JNZ     SPACES           ; Loop until enough spaces

02AD 3E20            ENDTAB: MVI    A,SPC     ; Print the last space when
02AF C9              RET                     ; get to GOPRNT

;
;
; ***** SUBR MODULO *****
;
; Does division of 2 8-bit numbers: (Reg E)/(Reg D)
; Result in Reg H
; Remainder in Reg C = (Reg E) mod (Reg D)
;
02B0 210800          MODULO: LXI     H,00001000B
02B3 0E00            MVI     C,0
02B5 7B              NEWBIT: MOV    A,E
02B6 17              RAL
02B7 5F              MOV     E,A
02B8 79              MOV     A,C
02B9 17              RAL
02BA 92              SUB     D
02BB D2BF02          JNC     NOADD
02BE 82              ADD     D
02BF 4F              NOADD: MOV    C,A
02C0 3F              CMC
02C1 7C              MOV     A,H
02C2 17              RAL
02C3 67              MOV     H,A
02C4 2D              DCR     L
02C5 C2B502          JNZ     NEWBIT
02C8 C9              RET

;
;
; ***** SUBR PRCRLF *****
;
PRCRLF:              ; Prints a CR/LF pair

02C9 F5              PUSH    PSW
02CA AF              XRA     A                ; Zero accum
02CB 327D04          STA     COLCNT           ; Set column counter to zero
02CE 3A7E04          LDA     LINECNT
02D1 3C              INR     A                ; Increment line counter
02D2 327E04          STA     LINECNT
02D5 3E0D            MVI     A,CR
02D7 CD2E03          CALL    PCHAR
02DA 3E0A            MVI     A,LF
02DC CD2E03          CALL    PCHAR
02DF F1              POP     PSW
02E0 C9              RET

;
;
; ***** SUBR BCDTOASC *****
;

```



```

; Converts binary value into two ASCII-hex characters
;
; ENTRY:  Reg A  = 2 packed-BCD characters
;          DE    = Adr of where to store ASCII answers
;
; EXIT:   (DE) = MS char
;          (DE+1) = LS char
;
BCDTOASC:
02E1 F5      PUSH    PSW
02E2 0F      RRC          ; Put 4 MSB's into 4 LSB's
02E3 0F      RRC
02E4 0F      RRC
02E5 0F      RRC
02E6 E60F    ANI      0FH      ; Mask out top 4 bits
02E8 C630    ADI      '0'      ; Add offset to make ASCII
02EA 12      STAX     D        ; Save the result
02EB F1      POP      PSW      ; Get the next character
02EC E60F    ANI      0FH
02EE C630    ADI      '0'      ; Make ASCII
02F0 13      INX      D
02F1 12      STAX     D        ; Save it in next memory loc
02F2 C9      RET

;
; ***** SUBR NTOLIT *****
;
; Converts packed-BCD number (1 to 12) to its equiv
; literal string and saves in specified memory.
;
; ENTRY:  A  = 2 packed-BCD numbers from clock
;          DE = Adr of where to store literal results
;          HL = Pntr to 10-char literal strings
;
02F3 47      NTOLIT: MOV    B,A      ; Save data
02F4 E610    ANI      00010000B    ; See if have tens digit
02F6 CAFF02  JZ       SMALLNR      ; Jump if not
02F9 78      MOV      A,B
02FA C60A    ADI      10           ; Add 10 to low nibble so
02FC C30003  JMP      BINARY      ; it's binary now.
02FF 78      SMALLNR:MOV    A,B

0300 E60F    BINARY: ANI      0FH      ; Mask off top nibble
0302 D601    SUI      1           ; Make number 0 to 11 max
0304 CA0F03  JZ       MOVE        ; At first already

0307 010A00  LXI      B,10        ; 10 literals in word
030A 09      NEXT:   DAD      B      ; Add it to HL for next word
030B 3D      DCR      A           ; until get to proper one.
030C C20A03  JNZ      NEXT

030F 010A00  MOVE:   LXI      B,10        ; Cnt to move 10 letters only

0312 7E      MOVDAT: MOV    A,M      ; HL pts to desired wor
0313 12      STAX     D           ; DE pts to destination mem.
0314 13      INX      D
0315 23      INX      H
0316 0D      DCR      C           ; Proper literal word now
0317 C21203  JNZ      MOVDAT      ; moved to memory to print

031A C9      RET

;
; ***** SUBR LISTIT *****
;
; Lists to printer until '$' encountered.
;
; ENTRY:  HL = string address
;
031B 7E      LISTIT: MOV    A,M      ; Get the letter to be sent
031C FE24    CPI      '$'          ; End of text yet?
031E CA2D03  JZ       LISTEND      ; Must retain HL!
0321 E5      PUSH    H
0322 5F      MOV      E,A
0323 0E05    MVI      C,LISTOUT
0325 CD0500  CALL     BDOS          ; Print the char

```

(Continued on next page)

Listing Three

```

0328 E1          POP      H
0329 23          INX      H
032A C31B03      JMP      LISTIT          ; Keep going until '$'
032D C9          LISTEND:RET
;
;
; *****          SUBR PCHAR          *****
;
PCHAR:          ; Prints Reg A to console or printer

IF TOLPTR
032E 0E05      MVI      C,LISTOUT          ; Output to printer
ENDIF
;
IF NOT TOLPTR
MVI      C,CONSOUT          ; Output to console
ENDIF

0330 5F          MOV      E,A
0331 CD0500      CALL     BDOS
0334 C9          RET
;
;
; *****          SUBR PRINTIT          *****
;
ENTRY:          DE = Start address of string
;
0335 F5          PRINTIT:PUSH     PSW

0336 0E09      MVI      C,PRINTST
0338 CD0500      CALL     BDOS          ; Print string to console
033B F1          POP      PSW
033C C9          RET
;
;
; *****          CONSOLE MESSAGES          *****
;

033D 0D0A496E70 OPENERR: DB      CR,LF,'Input file not found or not '
035B 7370656369 DB      'specified! ',CR,LF,'$'
;
;
; *****          TIME/DATE/PGM HEADING          *****
;

0368          TOPLINE: DS      SKIP

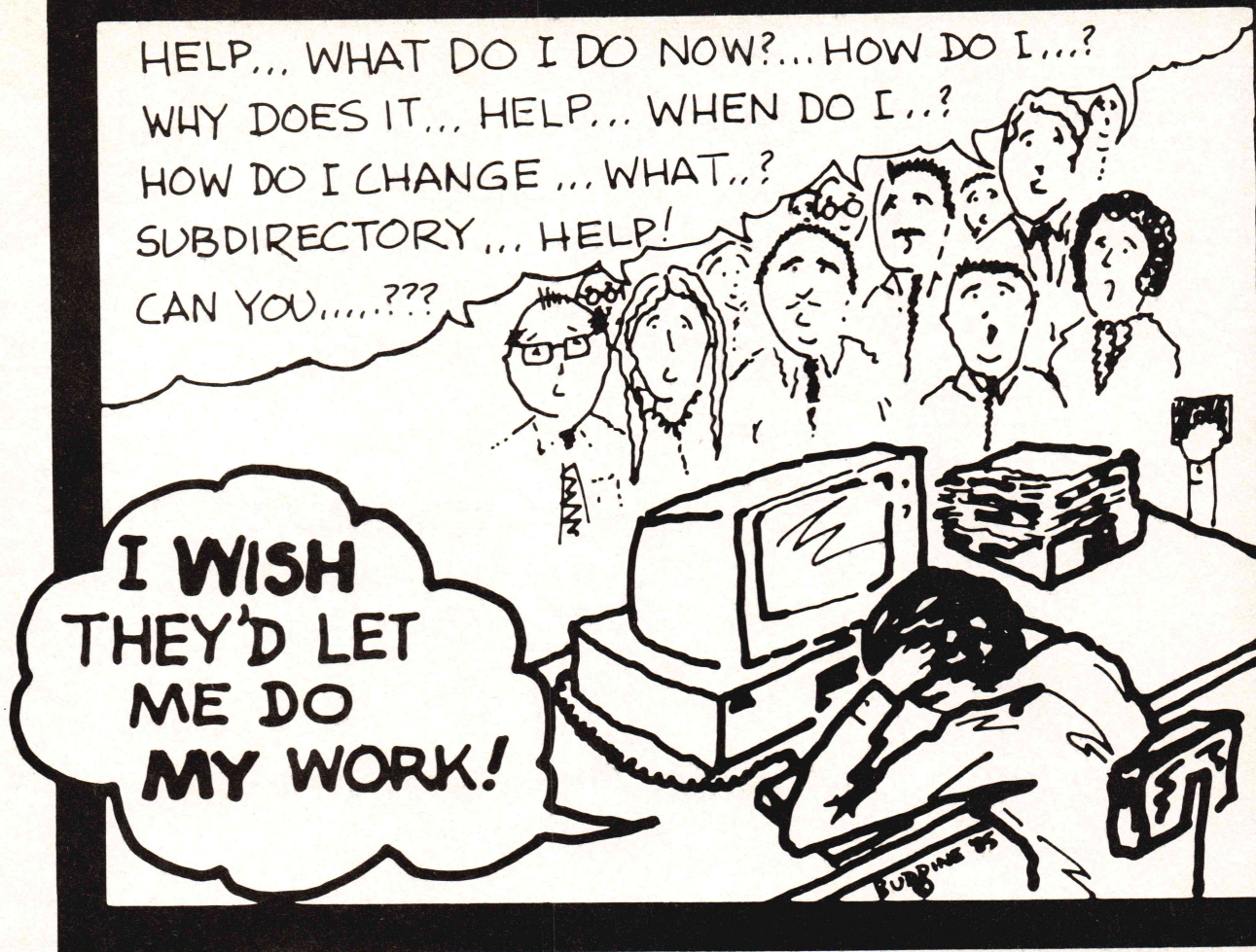
036B          HOURS:   DS      2
036D 3A          DB      ':'
036E          MINUTES: DS      2
0370 3A          DB      ':'
0371          SECONDS: DS      2
0373 2020202020 DB      ' '
0380          DAY:     DS      10
038A 20          DB      ' '
038B          DATE:    DS      2
038D 20          DB      ' '
038E          MONTH:   DS      10
0398 3139          DB      '19'
039A          YEAR:    DS      2
039C 2020202020 DB      ' '
03AA          HEADER:  DS      TSIZE
03B6 0D0A0A0A24 DB      CR,LF,LF,LF,'$'
;
;
; *****          LITERALS          *****
;

03BB 53756E6461 LITDAY: DB      'Sunday, '
03C5 4D6F6E6461 DB      'Monday, '
03CF 5475657364 DB      'Tuesday, '
03D9 5765646E65 DB      'Wednesday, '
03E3 5468757273 DB      'Thursday, '
03ED 4672696461 DB      'Friday, '

```

(Continued on page 90)

TIME IS MONEY



Your Wish Is Our **MenuCommand™**

Here is the perfect productivity tool for use by unsophisticated end users - or for integration into your own application system.

MenuCommand is a menu display editing system for PC/MS-DOS written in PC Forth. It is extremely fast and more powerful in features than any other menu system available. It supports unlimited menus and up to 24 selections per menu. Any menu selection can be specified to run a .COM or .EXE program (with automatic or screen-prompted parameters), any batch file, or any DOS command string up to 60 characters (with automatic or screen-prompted parameters). MenuCommand currently supports both cross-device and cross/sub-directory file access. Includes password protection of menus, DOS access, Menu Editor access, and menu selections (10 levels), and supports custom-color menu displays.

\$49.⁹⁵

Money-Back Guarantee
One Hour Free
Support Included



Command Software Systems, Inc.
5308 Derry Avenue, Suite K
Agoura Hills, CA 91301
(818) 707-7100

Circle no. 38 on reader service card.

Real-Time Clock (Listing Continued, text begins on page 56)

Listing Three

```

03F7 5361747572      DB      'Saturday, '
0401 4A616E7561 LITMON: DB      'January  '
040B 4665627275      DB      'February '
0415 4D61726368      DB      'March    '
041F 417072696C      DB      'April    '
0429 4D61792020      DB      'May      '
0433 4A756E6520      DB      'June     '
043D 4A756C7920      DB      'July     '
0447 4175677573      DB      'August   '

0451 5365707465      DB      'September'
045B 4F63746F62      DB      'October  '
0465 4E6F76656D      DB      'November '
046F 446563656D      DB      'December '

```

```

;
; *****      VARIABLES      *****
;

```

```

0479      OLDSTK:  DS      2              ; Stack pointer to get back
                                           ; to the CCP from this pgm.

047B      INBUFFT:  DS      2              ; Input buffer pointer

047D      COLCNT:   DS      1              ; Current length of line

047E      LINECNT:  DS      1              ; Keep count of lines printed
                                           ; on the page

```

```

;
; *****      STACK LOCATION      *****
;

```

```

047F      DS      32              ; Reserve 16-level stack

049F      NEWSTK:

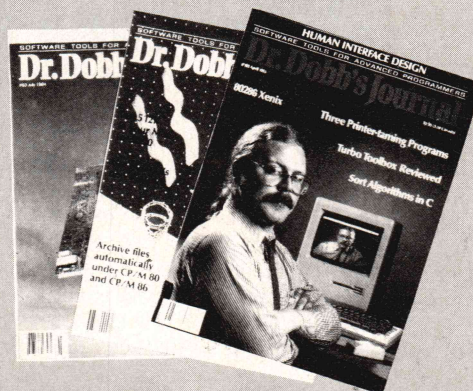
END

```

End Listings

AVAILABLE DDJ BACK ISSUES

1982	1983	1984	1985
No. 68—June	No. 76—Feb.	No. 87—Jan.	No. 99—Jan.
No. 69—July	No. 77—March	No. 88—Feb.	No. 100—Feb.
No. 70—Aug.	No. 78—April	No. 90—April	No. 101—March
No. 71—Sept.	No. 80—June	No. 91—May	No. 102—April
No. 72—Oct.	No. 81—July	No. 92—June	No. 103—May
No. 73—Nov.	No. 82—Aug.	No. 94—Aug.	No. 104—June
	No. 83—Sept.	No. 95—Sept.	
	No. 84—Oct.	No. 96—Oct.	
	No. 85—Nov.	No. 97—Nov.	
	No. 86—Dec.	No. 98—Dec.	



TO ORDER: send \$3.50 per issue to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303.

Name _____

Address _____

City _____

State _____ Zip 105

PRESENTING THE MEGAMAX C COMPILER

FEATURES:

- IN-LINE ASSEMBLY • ONE PASS COMPILATION • SUPPORT OF DYNAMIC OVERLAYS • FULL ACCESS OF MACINTOSH TOOLBOX ROUTINES • AND MUCH MORE...

DEVELOPMENT SYSTEM PACKAGE INCLUDES:

- FULL-SCALE IMPLEMENTATION (K&R) C COMPILER • THE STANDARD C LIBRARY • ROM ROUTINES LIBRARY • LINKER • LIBRARIAN AND DOCUMENTATION...

\$299.95

FOR MORE INFORMATION OR TO ORDER CALL OR WRITE:

DuVall Inc.
 BOX 851521 DEPT. Y
 RICHARDSON, TX
 75085-1521
 (214) 987-4937

MACINTOSH IS A
 REGISTERED TRADEMARK
 OF APPLE COMPUTER INC.

NOW AVAILABLE FOR THE MACINTOSH

Circle no. 84 on reader service card.

A Professional Quality Z80/8080/8085 Disassembler

WHEN YOU NEED SOURCE FOR YOUR CODE you need REVAS 3

REVAS interactively helps you:

Analyse your software for modification
disassemble files as large as 64K

Assign Real labels in the disassembly

Insert COMMENTS in the disassembly

Generate a Cross Reference (XREF) listing

A 60 page manual shows how the powerful REVAS command set gives you instant control over I/O to files, printer, or console; how to do a disassembly; and even how the disassembler works! You get on line help, your choice of assembler mnemonics, control of data interpretation, and calculation in any number base!

REVAS runs in Z80 CPM computers; is available on 8" SSSD (standard), RAINBOW, and other (ask) formats

Price: \$90.00 (plus applicable tax), Manual only: \$15.00

REVASCO

6032 Chariton Ave., Los Angeles, CA 90056

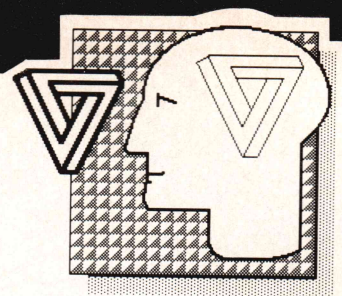
Voice: (213) 649-3575 Modem: (213) 670-9465

Circle no. 80 on reader service card.

**FOR THE BEST C DEVELOPMENT
SYSTEM AVAILABLE FOR THE MACINTOSH,
CALL DUVALL.
415-322-2757**

Consulair Corp.

Circle no. 57 on reader service card.



by Michael Swaine and Bob Albrecht

We announced this column of imagined and realizable projects in February in a piece called "Tiny Hackers" and actually launched it in March with Richard Stallman's "GNU Manifesto." But the point of writing about realizable fantasies is to encourage their realization, so this month we present progress reports on the fantasies presented to date.

Liberating the Mac

In January we published, though not in this column, Tom Lafleur's illustrated guide to fattening your Macintosh. Tom received over a hundred calls and letters shortly after the issue came out. By the time of the Macworld show, several people had set up business dedicated to Mac-fattening, at least some of which seem to have been spin-offs from Tom's article. Despite our warnings that doing it yourself voided your warranty, risked frying your Mac and was a tedious process, a lot of you opened your Macs and went at it. We think that one reason for the article's popularity is that a lot of you believe that Apple's decision to build the Mac as a closed system was a mistake, and one you're willing to rectify. ("Apple just ain't the company it once Woz."—Laran Stardrake.)

In April we followed up on this belief by reporting in this column on various ways in which people could be said to be liberating the Mac, including Lee Felsenstein's Hacker's Mac (aka Hackintosh), a project for learning about the Mac by redesigning it; and Jack Tramiel's Atari ST (aka Jackintosh), which, running DRI's GEM environment, may bring Maclike capabilities to the rest of us. As this goes to press, Atari has just previewed the ST in Germany, DRI has announced the first applications

for GEM and Lee thinks that Tramiel may be doing some of the Hackintosh team's work for it, though he hasn't abandoned the project. Contact Lee at Golemics, 2600 Tenth Street, Berkeley CA 94710, (415) 486-8344.

Also in April, we described Steve Jasik's MacNosy disassembler for the Mac, but failed to give a full address. It's Free the ROM 64, 343 Trenton Way, Menlo Park CA 94025, and his telephone number is (415) 322-1386. Steve has fixed some bugs in Nosy and is now supplying a fast sort routine with the program. MacNosy's opening screen echos the opening lines of the cult classic television program *The Prisoner*:

Who are you?	I am number two.
Who is number one?	You are number six.
What do you want?	Information.

GNU Manifestations

As last month's letters indicated, reaction to March's Realizable Fantasy, a proposal by Richard Stallman to develop a free operating system that provides the capabilities of Unix, has been emotional. Stallman's mail has been overwhelmingly supportive; he's received many offers of help, and he called to offer us quarterly updates on the project's progress. We'll probably print the first of these next month. Stallman is reachable at 166 Prospect St., Cambridge MA 02139.

Tiny Hackers

Dragonsmoke, Bob Albrecht's newsletter, is the place to follow The Dragon's master plan to turn innocent children into programmers, and it seems that it will soon be carrying excerpts from Ron Jeffries' newsletter, the *Jeffries Report*. Dragonsmoke costs

\$12/year and you can get a sample issue by sending \$1.00 or \$.39 and an SASE to Dragonsmoke, P.O. Box 7627, Menlo Park CA 94026.

Check the June issue of *Rainbow* for a review of the CoCoMac, a low-budget way to get Maclike features. The same issue also contains information on CoCoMac RAM disk construction and a plan to get the Fujitsu dual-6809 machine distributed in the U.S. It's supposedly very powerful, even without the 68K you can drop in.

Subversives

In a guest essay in this space in May, Resident Intern Dave Cortesi gave his vision of this magazine, a realizable fantasy for *DDJ*: that *DDJ* is and should always be subversive. He said, "As a good revolutionary, I must believe that computer use is for everybody, and I really do. But programming the computers so they can be used is almost certainly a job for specialists. Whose specialists?" Not the establishment's, Cortesi hopes, but those who want to promulgate ideas to all who can grasp them, those who do not see information exchange as a zero-sum game, those who see themselves involved in a revolution, those with a mischievous, subversive desire "to snatch the tools of the establishment and apply them in the public domain." We never ignore Cortesi's advice, and we hope you won't either. You have, through your contributions, the power to steer this magazine away from the safe shores and keep it pushing out recklessly on journeys of discovery.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 195.

CP/M SYSTEM INVENTORY LIQUIDATION



Old operating systems never die, they get better with age! Multitech Electronics, a large manufacturer of complete computer systems has an inventory of CP/M computers that must go. These fully guaranteed high performance units are new factory packaged systems with the CP/M 2.2 operating system, 64K of RAM, dual floppy disk drives, and CBASIC programming language. You'll get more than what you paid for with complete systems at bargain prices.

FULL CP/M SYSTEM SPECS

	MIC-501	MIC-504
CPU	Z-80A	Z-80A
RAM MEMORY	64K	64K
STORAGE	500K BYTES	2M BYTES
SERIAL COMMUNICATIONS	DUAL RS-232C PORTS	DUAL RS-232C PORTS
PARALLEL COMMUNICATIONS	SINGLE PARALLEL PORT	SINGLE PARALLEL PORT
PRICE	\$500	\$700

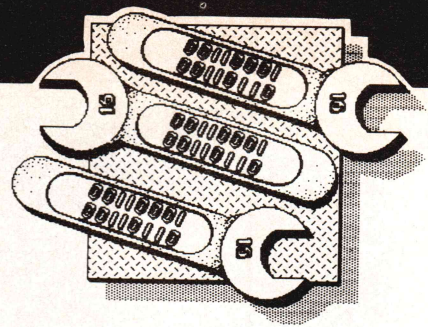
This low price also includes: full compatibility with the wide bank of CP/M software, CP/M 2.2 operating system with utilities and manuals, CBASIC 2 programming language, as well as a User's Manual and Service Manual. In addition you'll get a 30 day full warranty. And Multitech will also throw in a Liberty Freedom 100 terminal for \$400.00 with the purchase of an MIC-500

system. Call Multitech today for more on your CP/M solutions. Cash or credit cards only. Limited quantities available.



Multitech ELECTRONICS
INC.

195 WEST EL CAMINO REAL
SUNNYVALE, CALIFORNIA 94087
TELEX: 755042 MAC SUVL
IN CALIFORNIA, CALL: (408) 773-8400
OUTSIDE CALIFORNIA, CALL: (800) 538-1542



by Ray Duncan

MSDOS Installable Device Drivers

One of the more novel features added in Version 2 of MSDOS is the concept of "installable device drivers." This allows the user to attach new drivers for additional hardware devices or to supersede the system's existing built-in drivers by the simple expedient of putting the executable driver file on the boot disk and editing a line into the file CONFIG.SYS. This extremely powerful concept has made the lives of third-party mass storage device manufacturers much easier (previously they had to disassemble and patch the operating system to get their products to run).

But First . . . an Overview of Unix Device Drivers

Because the installable device drivers of MSDOS are patterned after Unix and much of the terminology used in the Microsoft driver documentation derives from Unix terminology, it is instructive to review the structure of the real McCoy. Much of the information presented here was gleaned from the article "Writing Device Drivers for Xenix Systems" by Jean McNamara, et al. (UniForum Conference Proceedings, January 1984).

Unix knows two types of devices: block and character. A block device is typically a mass storage medium such as a fixed or removable disk or magnetic tape drive. Such devices usually transfer data in chunks of fixed size, which are related (in the case of disks) to the characteristics and "format" of the physical media.

In contrast, a character device is a sequential device such as a CRT terminal or paper tape reader that supplies or accepts a stream of bytes. Although ideally the data stream of a character device has no inherent structure, in practice it usually is delimited by special control characters such as carriage returns, which the operating system also recognizes.

Under Unix, all drivers (whether controlling a block or character device) have the same general structure and contain two major parts: task time routines and an interrupt handler.

The task time routines are called at the time of the application program's (the task's) request for I/O. The Unix kernel transforms the request from its high-level, logical, device-independent character into addresses and parameters that are relevant to the physical device. When the driver's task time routines are executing, the application itself may be thought of as being in control of the system and active, although it is running in privileged or kernel mode.

The interrupt handler is entered asynchronously when the corresponding device generates a hardware interrupt. Interrupts are issued when an I/O operation has either finished or been terminated due to a hardware fault. If the system or the device does not support hardware interrupts, the interrupt handler is entered when the kernel, polling the device periodically, determines that it is no longer busy. Typically, the task requesting I/O from the device is inactive when the interrupt handler receives control; indeed, the task is usually in a suspended state pending I/O completion, and some other application is active and in control as far as the operating system is concerned.

A Unix block device driver comprises some tables and five major routines:

- *Init* is called once when the system is booted and the driver is first loaded into memory. It checks for the existence of the physical device and initializes it properly for future I/O, if present.
- *Open* is called by the Unix kernel when a user tries to access a file on the device. This routine should complete any initialization not performed by *Init* to prepare for I/O; it may be part of a mount sequence for removable media.
- *Close* is called by the Unix kernel to reset any flags or variables set by *Open* and to ensure that pending operations are completed. A typical action would be to flush buffered data to the physical device. This routine may be part of a dismount sequence for removable media.
- The *Strat* routine, an abbreviation for "strategy," is called by the Unix kernel when the application issues an I/O request for the block device. *Strat* is called with the address of a buffer header or parameter list, which defines the device unit number, request type (read, write, or format), where to find the data (logical block number), how much data it should transfer, and the memory address of its buffer. *Strat*'s responsibility is to validate the request then place it on the request queue for that device. Finally, it calls the internal routine *Start* (see below).
- *Intr* is called by the Unix kernel when the block device issues an interrupt. It is responsible for determining that the interrupt signal is valid (Was an operation on this device really in progress and are the appropriate completion flags set on the controller?). It determines success or failure status for the last operation by interrogating the controller and passes the appropriate flags back to the kernel. Finally, if additional requests are pending, it calls *Start* to try to initiate another I/O operation.

Start is an internal driver routine called by *Strat* or *Intr*; it removes the highest priority request from the request queue and initiates the I/O. If the device is busy or if no requests are waiting, *Start* simply exits.

The body of a Unix character device driver is similar to that of a block device driver. It is made up of seven major routines:

- *Init*, *Open*, and *Close* are similar in function and intent to those of the same name in a block device driver.
- *Read* transfers data from the device driver's input buffer to the application's buffer.
- *Write* transfers data from the application's buffers to the device driver's output buffers; it also starts output to the device if it is idle. If the output buffers are full, *Write* suspends the requesting process until they have emptied.
- *Ioctl* provides direct communication between the application and the driver. It allows the application to ask for device-specific information or to set the values of the driver's internal flags and variables (such as baud rate or number of stop bits for a serial I/O port).
- *Intr*, the asynchronous part of the driver, is called by the kernel when the device issues a hardware interrupt. This procedure performs the actual data transfer between the physical device's hardware controller and the driver's input/output buffers.

MSDOS Device Drivers

MSDOS installable device drivers bear a strong resemblance to Unix device drivers, both structurally and functionally. Like Unix, they fall into two major classes.

Character device drivers control devices that perform I/O one byte at a time, similar to a traditional TTY terminal. MSDOS has built-in drivers for the console device, serial port, and list device, named CON, AUX, and PRN, respectively. You can access these drivers using the traditional CP/M-like character I/O calls, or you can open them by name, like a file for input and output, using the new "handle" function calls of MSDOS 2.0 and above. A character device driver can support one hardware unit.

Block device drivers control random access storage devices such as flexible disk drives or fixed disks. A block

device driver can support more than one hardware unit and/or may map a single physical unit onto two or more logical drives. Each device unit for a given block driver is assigned a drive designator (such as A:, B:, etc.); the first drive letter for a given block device driver is determined by that driver's position in the overall chain of drivers.

Once a driver is written and assembled, you may load and link it into the operating system simply by placing an entry of the form

```
device=filename.ext
```

in the CONFIG.SYS file on the system boot disk. You can override the default system driver for a character device with an installed driver simply by giving it the same logical device name in the device header. When processing an I/O request, DOS always scans the list of installed drivers before the default devices and takes the first match.

Structure of an MSDOS Device Driver

MSDOS device drivers always have an ORIGIN of zero but are otherwise assembled, linked, and converted into an executable module as though they were COM or EXE files. A device driver consists of three major parts.

A Device Header (Table 1, below) contains the linkage to the next driver in the chain, a set of attribute flags (Table 2, below) for the device, offsets to the executable strategy and interrupt routines for the device, and the logical device name (if it is a character device such as PRN or COM1). The linkage address of the last driver in a file is

Offset	Contents
0	Offset, pointer to next device header
2	Segment, pointer to next device header
4	Device attribute word (see Table 2)
6	Pointer to device strategy code (offset)
8	Pointer to device interrupt code (offset)
10	Logical name (8 bytes) if character device or number of units (1 byte) if block device followed by 7 bytes that may contain a name or nothing at all

Table 1
Device driver header.

Bit	Meaning
15	= 1 if character device = 0 if block device
14	= 1 if IOCTL is supported
13	= 1 if non-IBM format (block only)
12	#
11	= 1 if open/close/RM is supported*
10	#
9	#
8	#
7	#
6	#
5	#
4	#
3	= 1 if current clock device
2	= 1 if current NUL device
1	= 1 if current standard output device
0	= 1 if current standard input device
# Currently undefined and should be 0	
* DOS 3.0 and above only; should be 0 for DOS 2.x	

Table 2
Device attribute word, found in device driver header; only bits 11, 13, and 14 have significance on block devices.

always set to -1 when the driver is created; the drivers are chained together, and the linkage fields are updated by MSDOS at system initialization.

The Strategy Routine for the device is entered by DOS via a Far Call when the driver is first loaded and installed and whenever an application program issues an I/O request for the device. DOS uses ES:BX to pass the Strategy routine a double-word pointer to a data structure, which is called a Request Header; this structure contains information about the type of operation to be performed. According to MSDOS conventions, the Strategy code never actually performs an I/O operation but simply saves the pointer to the Request Header.

The last and most complex part of a device driver is the misnamed Interrupt Routine. This code implements the device driver proper; it performs the actual operation based on the function code and other information passed in the Request Header. Status and completion information may be passed back to DOS in the same Header.

When an I/O request is issued, the Interrupt Routine

```

;
; MS-DOS Request Header structure definition
;
Request  struc          ; request header template
                        structure
Rlength  db  ?          ; 0 length of request header
Unit     db  ?          ; 1 unit number for this request
Command  db  ?          ; 2 request header's command
                        code
Status   dw  ?          ; 3 driver's return status word
Reserve  db  8 dup (?)  ; 5 reserved area
Media    db  ?          ; 13 media descriptor byte
Address  dd  ?          ; 14 memory address for transfer
Count    dw  ?          ; 18 byte/sector count value
Sector   dw  ?          ; 20 starting sector value
Request  ends          ; end of request header template

```

Table 3

Format of request header. Only the first 13 bytes are common to all driver functions; the number and definition of the following bytes vary depending on the function type. The structure shown here is the one used by the read and write subfunctions of the driver.

Bit(s)	Significance
15	Error
10-14	Reserved
9	Busy
8	Done
0-7	Error code if bit 15 = 1

Table 4

Return status word of request header.

entry point is called by DOS immediately after the call to the Strategy Routine. Unlike in Unix, the Interrupt Routine is never entered asynchronously (as is the case on an I/O completion interrupt). The division of function between the Strategy and Interrupt Routines is completely artificial, at least under the currently available nonmultitasking versions of MSDOS.

The Request Header

The MSDOS BDOS uses a data structure called the Request Header to give the driver the necessary information to perform an I/O operation. The address of the Request Header is passed to the Strategy Routine and saved in a local variable; the Interrupt Routine, which is called immediately afterward, uses this address to access information from the Header.

The first 13 bytes of the Request Header are the same for all device driver functions and therefore are referred to as the "static" portion of the Header. The number and contents of the following bytes vary according to the type of function requested (Table 3, at left). The Request Header's primary components are a Command Code that selects a driver subfunction (such as Read, Write, or Status) and a Return Status word that informs the BDOS about the driver's success with the request I/O operation (Table 4, below left, and Table 5, page 98). Other information passed in the Header to the driver includes minor unit numbers, transfer addresses, sector or byte counts, and so on.

When an I/O function is completed, the device driver uses fields in the Request Header to pass status, sector or byte counts, and other information back to the operating system.

The Driver Command Code Routines

The Command Code for the requested driver subfunction is passed in the third byte of the Request Header. The Interrupt Routine extracts it from the Request Header using the double-word pointer saved during the call to the Strategy Routine. Typically, the Command Code is used as an index into a jump table that points to the proper service code.

In the descriptions below, RH refers to the Request Header whose address was passed to the Strategy Routine in ES:BX. DWORD refers to a long address, of which the first two bytes contain the offset and the last two bytes contain the segment.

Function 0—Driver Initialization

This initialization code for the driver is called only once, when the driver is loaded. It is responsible for performing any necessary hardware initialization of the device, setup of interrupt vectors, and so on. It returns the address of the start of free memory after the driver, so that DOS knows where it can load the next installable driver. If it is initializing a block device driver, it must also return the number of units and the address of the BIOS parameter block (BPB) pointer array; if all units are the same, all pointers in the array can point to the same BPB.

Technical Product Information

FREE

For The Asking

See something
you'd like to
learn more about?
Need more details?

DR. DOBB'S JOURNAL

Reader Service Card

Name _____ Phone _____

Address _____

City/State/Zip _____

Expiration Date: Oct. 31, 1985

July 1985 #106

Please circle one for each category:

I. My firm or department is a:

- A. Systems Integrator/House
- B. Software Dev. Firm
- C. Hardware OEM or Manuf.
- D. DP, MIS or Data Service
- E. Consulting Firm
- F. Eng. or Science Lab.
- G. Other

II. My job function is:

- H. Company Mgmt./Admin.
- J. Computer Systems Mgt.
- K. Programmer/Technical Staff
- L. Consultant
- M. Engineering Mgmt. or Staff
- N. Scientific Mgmt. or Staff
- O. Other

III. Number of employees in my firm:

- 1. Less than 10
- 2. 10—99
- 3. 100—499
- 4. 500—9,999
- 5. 10,000—or More

IV. This inquiry is for:

- P. Immediate Purchase
- Q. Future Project

V. Purchasing Authority

- (check all that apply)
- R. Recommend or Specify
- S. Final Decision-Maker
- T. No Influence

VI. I advise others about computers, on the average:

- 6. More Than Once-A-Day
- 7. Once-A-Day
- 8. Once-A-Week
- 9. Not At All

VII. I design/write software professionally

- U. Yes
- V. No

VIII. I buy computer products through:

- (check all that apply)
- W. Retail Stores
- X. Mail Order Houses
- Y. On-site Direct Salespeople
- Z. All of the Above

IX. I am currently a subscriber:

- A. Yes
- B. No

Check each advertisement for corresponding number and circle below:

001	011	021	031	041	051	061
002	012	022	032	042	052	062
003	013	023	033	043	053	063
004	014	024	034	044	054	064
005	015	025	035	045	055	065
006	016	026	036	046	056	066
007	017	027	037	047	057	067
008	018	028	038	048	058	068
009	019	029	039	049	059	069
010	020	030	040	050	060	070

071	081	091	101	111	121	131
072	082	092	102	112	122	132
073	083	093	103	113	123	133
074	084	094	104	114	124	134
075	085	095	105	115	125	135
076	086	096	106	116	126	136
077	087	097	107	117	127	137
078	088	098	108	118	128	138
079	089	099	109	119	129	139
080	090	100	110	120	130	140

141	151	161	171	Articles		
142	152	162	172	181	191	201
143	153	163	173	182	192	202
144	154	164	174	183	193	203
145	155	165	175	184	194	204
146	156	166	176	185	195	205
147	157	167	177	186	196	206
148	158	168	178	187	197	207
149	159	169	179	188	198	208
150	160	170	180	189	199	209
				190	200	210

☐ Please send me a one year subscription to Dr. Dobb's Journal at \$25.00 and bill me later.

Note:

For quicker, more effective processing
of your inquiry, please provide responses
to ALL requested information.

Information that's

- Current
- In-depth
- Directed to you on specific products & services

FREE

Send us your
POSTAGE-PAID card Today!

DR. DOBB'S JOURNAL

Reader Service Card

Name _____ Phone _____

Address _____

City/State/Zip _____

Expiration Date: Oct. 31, 1985

July 1985 #106

Please circle one for each category:

I. My firm or department is a:

- A. Systems Integrator/House
- B. Software Dev. Firm
- C. Hardware OEM or Manuf.
- D. DP, MIS or Data Service
- E. Consulting Firm
- F. Eng. or Science Lab.
- G. Other

II. My job function is:

- H. Company Mgmt./Admin.
- J. Computer Systems Mgt.
- K. Programmer/Technical Staff
- L. Consultant
- M. Engineering Mgmt. or Staff
- N. Scientific Mgmt. or Staff
- O. Other

III. Number of employees in my firm:

- 1. Less than 10
- 2. 10—99
- 3. 100—499
- 4. 500—9,999
- 5. 10,000—or More

IV. This inquiry is for:

- P. Immediate Purchase
- Q. Future Project

V. Purchasing Authority

- (check all that apply)
- R. Recommend or Specify
- S. Final Decision-Maker
- T. No Influence

VI. I advise others about computers, on the average:

- 6. More Than Once-A-Day
- 7. Once-A-Day
- 8. Once-A-Week
- 9. Not At All

VII. I design/write software professionally

- U. Yes
- V. No

VIII. I buy computer products through:

- (check all that apply)
- W. Retail Stores
- X. Mail Order Houses
- Y. On-site Direct Salespeople
- Z. All of the Above

IX. I am currently a subscriber:

- A. Yes
- B. No

Check each advertisement for corresponding number and circle below:

001	011	021	031	041	051	061
002	012	022	032	042	052	062
003	013	023	033	043	053	063
004	014	024	034	044	054	064
005	015	025	035	045	055	065
006	016	026	036	046	056	066
007	017	027	037	047	057	067
008	018	028	038	048	058	068
009	019	029	039	049	059	069
010	020	030	040	050	060	070

071	081	091	101	111	121	131
072	082	092	102	112	122	132
073	083	093	103	113	123	133
074	084	094	104	114	124	134
075	085	095	105	115	125	135
076	086	096	106	116	126	136
077	087	097	107	117	127	137
078	088	098	108	118	128	138
079	089	099	109	119	129	139
080	090	100	110	120	130	140

141	151	161	171	Articles		
142	152	162	172	181	191	201
143	153	163	173	182	192	202
144	154	164	174	183	193	203
145	155	165	175	184	194	204
146	156	166	176	185	195	205
147	157	167	177	186	196	206
148	158	168	178	187	197	207
149	159	169	179	188	198	208
150	160	170	180	189	199	209
				190	200	210

☐ Please send me a one year subscription to Dr. Dobb's Journal at \$25.00 and bill me later.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

P.O. BOX 13851

PHILADELPHIA, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

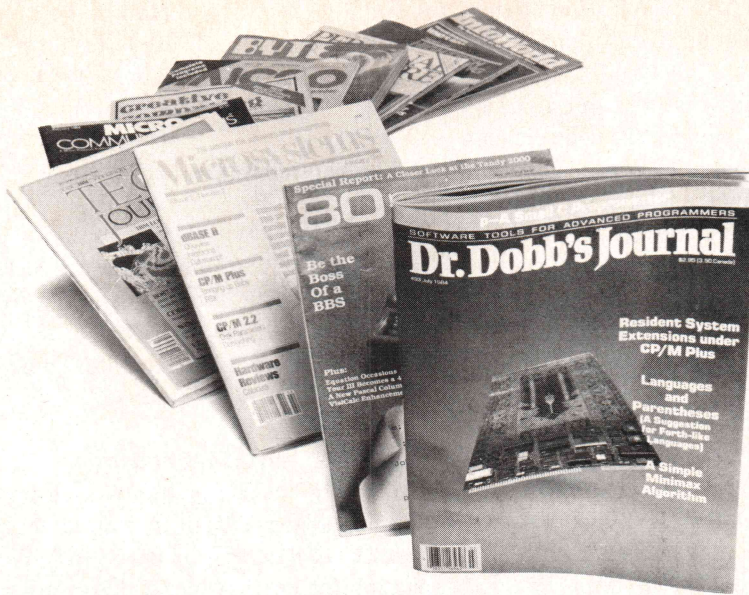
POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

P.O. BOX 13851

PHILADELPHIA, PA 19101



Dr. Dobb's Stands Apart

Take a good, hard look at the crowded computer magazine market. If you're a serious microcomputerist, one magazine stands apart. ***Dr. Dobb's Journal***.

Dr. Dobb's is not for everyone. It is written by and for expert programmers, and it's the oldest and most technically sophisticated microcomputer publication available. Since 1976, ***Dr. Dobb's Journal*** has been the unchallenged leading source of software tools for advanced programmers.

With the industry moving forward faster than ever, you need to stay a step ahead. ***Dr. Dobb's*** sets the pace with:

- regular columns on C, Unix, CP/M and 16-bit software;
- algorithms and problem solving;
- lively discussions of fundamental software issues;
- inside information on commercial languages and operating systems;
- tips on advanced programming topics.

The information and valuable code contained in ***Dr. Dobb's*** makes each issue indispensable for serious computing professionals and enthusiasts. Don't miss a single issue of this valuable resource. Subscribe today. If you aren't fully satisfied, cancel your subscription and keep the first issue as a free sample.

To start your subscription, fill out this coupon and return it to:

Dr. Dobb's Journal
P.O. Box 27809, San Diego, CA 92128

.....
Yes! Please enter my subscription for 12 issues of:
Dr. Dobb's Journal for \$25.

If I am not fully satisfied I will write "cancel" on my subscription invoice and keep the first issue as a free sample.

Name _____

Address _____

City _____

State _____ Zip _____

☐ Bill me later ☐ I've enclosed \$25 Charge my ☐ VISA ☐ M/C

Card No. _____

Expiration _____

Signature _____

Offer good in U.S. only

4005

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS
Dr. Dobb's Journal

If the initialization routine finds that the device is missing or defective and wants to abort without using memory, it should set the number of units to zero and set the ending address to CS:0000.

The operating system services that the initialization code can invoke at load time are limited; the code can call only MSDOS services 01-0CH and 30H. This is adequate to check the DOS version number and display a driver identification message, but not much else.

Many programmers position the initialization code at the end of the driver and return its address as the location of the first free memory; this allows the memory occupied by the code to be reclaimed after it is finished with its work.

This routine is called with:

RH+18 DWORD Pointer to the character after the = on the CONFIG.SYS line that

Code	Error Definition
0	Write protect violation
1	Unknown unit
2	Drive not ready
3	Unknown command
4	CRC error
5	Bad drive request structure length
6	Seek error
7	Unknown media
8	Sector not found
9	Printer out of paper
10	Write fault
11	Read fault
12	General failure
13-14	Reserved
15	Invalid disk change (MSDOS 3.x)

Table 5

Driver error codes returned in bits 0-7 of return status word in request header.

Byte(s)	Contents
0-1	Bytes per sector
2	Sectors per allocation unit (must be power of 2)
3-4	Number of reserved sectors (starting at sector 0)
5	Number of file allocation tables (FATs)
6-7	Maximum number of root directory entries
8-9	Total number of sectors in media
10	Media descriptor byte
11-12	Number of sectors occupied by a single FAT

Table 6

Format of BIOS parameter block; a copy of this block is always found in the boot sector of an initialized disk.

loaded driver; this information is read only

RH+22 BYTE Drive letter for first unit of a block driver: 0=A, 1=B, etc. (MSDOS 3.x only)

This routine returns:

RH+3 WORD Return status
 RH+13 BYTE Number of units (block devices only)
 RH+14 DWORD Address of first free memory above driver
 RH+18 DWORD BPB pointer array (block devices only)

Function 1—Media Check

The media check function is used on block devices only and should be a NOP in character device drivers. This routine is called first by BDOS for a block device transfer, passing the current media descriptor byte (Table 7, page 99). If feasible, the routine returns a code indicating whether the media has changed since the last transfer. This feature requires a machine that provides a door interlock hardware status flag or something similar. Determining the media change status improves performance because MSDOS does not need to reread the file allocation table (FAT) for each directory access.

This routine is called with:

RH+1 BYTE Unit code
 RH+13 BYTE Media descriptor byte

This routine returns:

RH+3 WORD Return status
 RH+14 BYTE Media change code:
 -1 Media has changed
 0 Don't know if media changed
 1 Media has not changed
 RH+15 DWORD Pointer to previous volume ID, if device attribute bit 11 = 1 and media has changed (MSDOS 3.x only)

Function 2—Build BIOS Parameter Block

The build BPB function is supported on block devices only and should be a NOP for character devices. The BDOS uses it to get a pointer to the valid BPB (Table 6, at left) for the current media. This routine is called when the media check routine returns a "Media has changed" code or when it returns a "Don't know if media changed" code and there are no dirty buffers (buffers with changed data that have not yet been written to disk). Thus this call indicates whether the media has legally changed. Under MSDOS 3.x, this function should also read the volume ID off the disk and save it.

The build BPB call also receives a pointer to a one-sector buffer in the address field of the request header. If the "non-IBM-format" bit in the device attribute word is

zero, the buffer contains the first sector of the FAT, including the media descriptor byte, and should not be altered by the driver. If the "non-IBM-format" bit is set, the buffer may be used as scratch space.

This routine is called with:

RH+1 BYTE Unit code
RH+13 BYTE Media descriptor byte
RH+14 DWORD Buffer address

This routine returns:

RH+3 WORD Return status
RH+18 DWORD Pointer to new BPB

Function 3—I/O Control Read

This function allows control information to be passed directly from the application program to the device driver. It is called only if the IOCTL bit is set in the device header attribute word. No error check is performed on IOCTL I/O calls.

This routine is called with:

RH+1 BYTE Unit code (block devices only)
RH+13 BYTE Media descriptor byte
RH+14 DWORD Transfer address
RH+18 WORD Byte/Sector count
RH+20 WORD Starting sector number (block devices only)

This routine returns:

RH+3 WORD Return status
RH+18 WORD Actual bytes or sectors transferred

Function 4—Read

This function transfers data from the device into the specified memory buffer. Under MSDOS 3.x, the routine can use the reference count of open files maintained by the open and close routines (functions 13 and 14) and the media descriptor byte to determine whether the media has changed illegally.

This routine is called with:

RH+1 BYTE Unit code (block devices only)
RH+13 BYTE Media descriptor byte
RH+14 DWORD Transfer address
RH+18 WORD Byte/Sector count
RH+20 WORD Starting sector number (block devices only)

This routine returns:

RH+3 WORD Return status
RH+18 WORD Actual bytes or sectors transferred
RH+22 DWORD Pointer to volume ID if error 0FH is returned (MSDOS 3.x only)

Function 5—Nondestructive Read

This function is available on character devices only. If an input status request returns a busy bit = 0 (characters waiting), the next character that would be read is returned to DOS but stays in the input buffer. This basically provides DOS with the capability to look ahead by one character.

This routine returns:

RH+3 WORD Return status
RH+13 BYTE Character

Function 6—Input Status

This function is available on character devices only and returns the current input status for the device. MSDOS

Media Descriptor Byte

Bit(s)	Significance
3-7	Always set (= 1)
2	1 = removable 0 = not removable
1	1 = 8 sector 0 = not 8 sector
0	1 = 2 sided 0 = not 2 sided

Current Valid MSDOS Descriptor Bytes (5¼-inch Disks)

0F9H	2 sided, 15 sector
0FCH	1 sided, 9 sector
0FDH	2 sided, 9 sector
0FEH	1 sided, 8 sector
0FFH	2 sided, 8 sector
0F8H	(fixed disk)

Table 7

Media descriptor byte of request header, assuming "non-IBM-format" bit in attribute word of device header is zero.

Addr	Attr	Str	Int	Type	Units	Name
00E3:0111	8004	0FD5	0FE0	C		NUL
0070:0148	8013	008E	0099	C		CON
0070:01DD	8000	008E	009F	C		AUX
0070:028E	8000	008E	00AE	C		PRN
0070:0300	8008	008E	00C3	C		CLOCK
0070:03CC	0000	008E	00C9	B	02	
0070:01EF	8000	008E	009F	C		COM1
0070:02A0	8000	008E	00AE	C		LPT1
0070:06F0	8000	008E	00B4	C		LPT2
0070:0702	8000	008E	00BA	C		LPT3
0070:0714	8000	008E	00A5	C		COM2
End of device chain.						

Table 8

Example listing of device chain under MSDOS 2.1: "plain vanilla" PC with no hard disks or user device drivers.

assumes all character devices have a type-ahead buffer. If the device does not have a type-ahead buffer, it should always return a busy bit = 0 so MSDOS will not hang.

This routine returns:

RH+3 WORD Return status
Busy bit = 1 Read request goes to physical device
Busy bit = 0 Characters already in device buffer; read request returns quickly

Function 7—Flush Input Buffers

This function is available on character devices only. It terminates all pending requests; i.e., the input buffer is emptied.

This routine returns:

RH+3 WORD Return status

Function 8—Write

This routine transfers data from the specified memory buffer to the device. Under MSDOS 3.x, the routine can use the reference count of open files maintained by the open and close routines (functions 13 and 14) and the media descriptor byte to determine whether the media has changed illegally.

This routine is called with:

RH+1 BYTE Unit code (block devices only)
RH+13 BYTE Media descriptor byte
RH+14 DWORD Transfer address
RH+18 WORD Byte/Sector count
RH+20 WORD Starting sector number (block devices only)

This routine returns:

RH+3 WORD Return status
RH+18 WORD Actual bytes or sectors transferred
RH+22 DWORD Pointer to volume ID if error 0FH is returned (MSDOS 3.x only)

Function 9—Write with Verify

This routine transfers data from the specified memory buffer to the device. If feasible, a read-after-write verification of the data is performed to confirm that the data was written correctly. Otherwise, it is exactly like function 8.

Function 10—Output Status

This routine returns the current output status for the device. This function is available on character devices only.

This routine returns:

RH+3 WORD Return status
Busy bit = 1 Write request waits

Language & Compiler Designers,

Use State-Of-The-Art
Parsing Technology With Our . . .

LALR (1) Grammar Analyzer & Parser Table Generator

it's . . .

- **Small.** Runs on IBM PC in 256K.
- **Fast.** Processes 500 state grammars in 1 min., 1000 states in 3 min.
- **Guaranteed.** 100% refund if not satisfied within 30 days.

by . . .

PAUL MANN
450 E. First St., Suite B-300
Tustin, CA 92680

714-771-9530

for only . . .

\$50

Check or
Money Order

California Residents Add 6% Sales Tax

Circle no. 60 on reader service card.

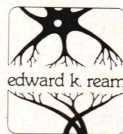
C Source Code

RED

Full Screen Text Editor

IBM PC, Kaypro, CP/M 80 and CP/M 68K systems.

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically and quickly.
 - RED is easy to use for writers or programmers. RED's commands are in plain English.
 - RED comes with complete source code in standard C. RED has been ported to mainframes, minis and micros.
 - RED comes with a Reference Card and a Reference Manual that provides everything you need to use RED immediately.
 - RED is unconditionally guaranteed. If for any reason you are not satisfied with RED your money will be refunded promptly.
- RED: \$95**
Manual: \$10



Call or write today for
for more information:

Edward K. Ream
1850 Summit Avenue
Madison, WI 53705
(608) 231-2952

To order:

Either the BDS C compiler or the Aztec CII compiler is required for CP/M80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format described (8 inch CP/M single density or exact type of 5 1/4 inch disk).

Send a check or money order for \$95 (\$105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited.

Circle no. 13 on reader service card.

for completion of current request
Busy bit = 0 Device idle; write request starts immediately

Function 11—Flush Output Buffers

This function is available on character devices only and terminates all pending output requests. The output buffer, if any, is emptied.

This routine returns:

RH+3 WORD Return status

Function 12—I/O Control Write

This function allows control information to be passed directly from the driver to the application program. It is called only if the IOCTL bit is set in the device header attribute word. No error check is performed on IOCTL I/O calls.

This routine is called with:

RH+1 BYTE Unit code (block devices only)
RH+13 BYTE Media descriptor byte
RH+14 DWORD Transfer address
RH+18 WORD Byte/Sector count
RH+20 WORD Starting sector number (block devices only)

This routine returns:

RH+3 WORD Return status
RH+18 WORD Actual bytes or sectors transferred

Function 13—Open

This function is available on MSDOS version 3.0 and above only and is called only if the open/close/RM bit is set in the device attribute word.

Block devices may use the open function to manage local buffering and to increment a reference count of the number of open files on a device.

Character devices can use this function to send a device initialization string, which in turn can be set by IOCTL Write. Note that the predefined CON, AUX, and PRN devices are always open.

This routine is called with:

RH+1 BYTE Unit code (block devices only)

This routine returns:

RH+3 WORD Return status

Function 14—Device Close

This function is available on MSDOS version 3.0 and above only and is called only if the open/close/RM bit is set in the device attribute word.

On block devices, this function can manage local buffering and decrement a reference count keeping track of the number of open files on the device; when the count reaches zero, all files have been closed, and the driver should flush buffers because the user may change disks.

On character devices, this function can send a device-dependent post-I/O string such as a form feed, which in turn can be set by an IOCTL Write. Note that the CON, AUX, and PRN devices are never closed.

This routine is called with:

RH+1 BYTE Unit code (block devices only)

This routine returns:

RH+3 WORD Return status

Function 15—Removable Media

This function is available on MSDOS version 3.0 and above only and is called only if the open/close/RM bit is set in the device attribute word and the device is a block type.

This routine is called with:

RH+1 BYTE Unit code (block devices only)

This routine returns:

RH+3 WORD Return status
Busy bit = 1 Media is non-removable
Busy bit = 0 Media is removable

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 196.

16-Bit Toolbox Listing (Text begins on page 94)

```
1 name driver
2 page 55,132
3 title 'DRIVER --- installable driver template'
4
5 ;
6 ; This is a "template" for a MS-DOS installable device driver.
7 ; The actual driver subroutines are stubs only and have
8 ; no effect but to return a non-error "done" status.
9 ;
10 ; Ray Duncan, April 1985
```

(Continued on next page)


```

11                                     ; Laboratory Microsystems Inc.
12
13     0000                           code    segment public 'CODE'
14
15     0000                           driver  proc    far
16
17                                     assume  cs:code,ds:code,es:code
18
19     0000                           org      0
20
21     = 000F                         Max_Cmd equ    15                ; driver command code maximum
22                                     ; 12 for MS-DOS 2.x,
23                                     ; 15 for MS-DOS 3.x
24
25     = 000D                         cr      equ    0dh                ; ASCII carriage return
26     = 000A                         lf      equ    0ah                ; ASCII line feed
27     = 0024                         eom     equ    '$'                ; end of message signal
28
29                                     page
30
31                                     ;
32                                     ; Device Driver Header
33                                     ;
34     0000 FF FF FF FF               Header dd      -1                ; link to next device, -1= end of list
35
36     0004 8000                      dw        8000h                ; attribute word
37                                     ; bit 15=1 for character devices
38
39     0006 0056 R                    dw        Strat                ; device "Strategy" entry point
40
41     0008 0061 R                    dw        Intr                 ; device "Interrupt" entry point
42
43     000A 44 52 49 56 45 52         db        'DRIVER '            ; char device name, 8 char, or
44     20 20                                     ; if block device, no. of units
45                                     ; in first byte followed by
46                                     ; 7 don't care bytes
47
48
49                                     ;
50                                     ; local variables for use by driver
51                                     ;
52     0012 ????????                 RH_Ptr  dd      ?                ; pointer to request header
53                                     ; passed to Strat by BDOS
54
55     0016 0D 0A 0A                 Ident   db        cr,lf,lf
56     0019 45 78 61 6D 70 6C         db        'Example Device Driver 1.0'
57     65 20 44 65 76 69
58     63 65 20 44 72 69
59     76 65 72 20 31 2E
60     30
61     0032 0D 0A 0A 24              db        cr,lf,lf,eom
62
63                                     page
64
65                                     ;
66                                     ; Driver Command Codes dispatch table
67                                     ;
68                                     ; The "Intr" routine uses this table and the Command Code
69                                     ; supplied in the Request Header to transfer to the
70                                     ; appropriate driver subroutine.
71
72     0036                           Dispatch:
73
74     0036 00BA R                    dw        Init                ; 0 = init driver into system
75     0038 009C R                    dw        Media_Chk           ; 1 = media check on blk dev
76     003A 009E R                    dw        Build_Bpb           ; 2 = build BIOS param block
77     003C 00A0 R                    dw        Ioctl_Inp           ; 3 = I/O ctrl read from dev
78     003E 00A2 R                    dw        Input              ; 4 = normal destructive read
79     0040 00A4 R                    dw        Nd_Input           ; 5 = non-destructive read, no wait
80     0042 00A6 R                    dw        Inp_Stat           ; 6 = return current input status
81     0044 00A8 R                    dw        Inp_Flush          ; 7 = flush device input buffers
82     0046 00AA R                    dw        Output             ; 8 = normal output to device
83     0048 00AC R                    dw        Outp_Vfy           ; 9 = output with verify
84     004A 00AE R                    dw        Outp_Stat          ; 10 = return current output status

```

(Continued on page 104)

C Programmers: File System Utility Libraries

Source Code Included, No Royalties,
Powerful & Portable.

BTree Library

\$75.⁰⁰

- High speed random and sequential access.
- Multiple keys per data file.
- Up to 16 million records per file.
- Full documentation and example programs included.

ISAM Driver

\$40.⁰⁰

- Works with the BTree Library.
- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use; fully documented; example programs included.

Both products

Are written entirely in K&R C.
Come with complete source code. + \$3.00 Shipping & Handling Charge.
Are free of any royalty charges.

For more information call:

softfocus

Credit cards accepted.

1277 Pallatine Drive
Oakville, Ontario, Canada
L6H 1Z1
(416) 844-2610

Dealer inquiries invited.

Wizard C

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

- All UNIX System V language features
- Support for 8087, 80186 and 80286
- Full library source code included
- Cross-file checks (full UNIX lint)
- Uses MS-LINK or PLINK 86
- ROMable data options
- In-line assembly language
- Cross compilers available
- Third party software available, including PANEL

The new standard for C Compilers on MSDOS!

Only \$450.

(617) 641-2379

WIZARD

Systems Software, Inc.

11 Willow Court
Arlington, MA 02174



Circle no. 116 on reader service card.

EXTRA!

**VALUE and PERFORMANCE
with Relocatable Z-80
Macro Assembler
FROM MITEK**

It's a real bargain! Here's why:

- Only \$49.95 plus shipping
- 8080 to Z-80 Source Code Converter
- Generates Microsoft compatible REL files or INTEL compatible hex files
- Compatible with Digital Research macro assemblers MAC & RMAC
- Generates Digital Research compatible SYM files
- Full Zilog mnemonics
- INCLUDE and MACLIB files
- Conditional assembly
- Phase/dephase
- Separate data, program, common and absolute program spaces
- Customize the Macro Assembler to your requirements with installation program
- Cross-reference Generation
- Z-80 Linker and Library Manager for Microsoft compatible REL files available as a total package with Macro Assembler for only \$95.00 plus shipping
- Manual only is \$15

**NEW
for
Turbo Pascal
Users**
**Mitek's Relocatable Z-80
Macro Assembler will also:**

- Generate Turbo Pascal in-line machine code include files
- Include files provide Turbo Pascal compatible machine code with assembly language mnemonics as comments

TO ORDER, CALL TOLL FREE: 1-800-367-5134, ext. 804
For information or technical assistance: 1-808-623-6361

Specify desired 5 1/4" or 8" format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed. Include \$5 for postage and handling.

Z-80 is a trademark of Zilog, Inc. MAC, RMAC, and ZSID are trademarks of Digital Research, Inc.

MITEK

P.O. Box 2151,
Honolulu, HI 96805

Circle no. 66 on reader service card.


```

84      004C 00B0 R      dw      Outp_Flush      ; 11 = flush output buffers
85      004E 00B2 R      dw      Ioctl_Outp      ; 12 = I/O control output
86      0050 00B4 R      dw      Dev_Open      ; 13 = device open      (MS-DOS 3.x)
87      0052 00B6 R      dw      Dev_Close     ; 14 = device close     (MS-DOS 3.x)
88      0054 00B8 R      dw      Rem_Media     ; 15 = removeable media (MS-DOS 3.x)
89
90      page
91      ;
92      ; MS-DOS Request Header structure definition
93      ;
94      ; The first 13 bytes of the Request Header are the same
95      ; for all Command codes and are termed the "Static" part of
96      ; the Header. The number and meaning of the following bytes
97      ; vary depending on the Command code.
98      ;
99      ; The Request Header shown here applies to Read & Write functions.
100     ;
101
102     Request struc      ; request header template structure
103
104     0000 ??      Rlength db      ?      ; length of request header
105     0001 ??      Unit db      ?      ; unit number for this request
106     0002 ??      Command db      ?      ; request header's command code
107     0003 ???     Status dw      ?      ; driver's return status word
108     0005 08 [    Reserve db      8 dup (?) ; reserved area
109     ??          ]
110
111
112     000D ??      Media db      ?      ; media descriptor byte
113     000E ???????? Address dd      ?      ; memory address for transfer
114     0012 ???     Count dw      ?      ; byte/sector count value
115     0014 ???     Sector dw      ?      ; starting sector value
116
117     0016      Request ends      ; end of request header template
118
119     page
120
121     ; Device Driver "Strategy Routine"
122
123     ; Each time a request is made for this device, the BDOS
124     ; first calls "Strategy routine", then immediately calls
125     ; the "Interrupt routine".
126
127     ; The Strategy routine is passed the address of the
128     ; Request Header in ES:BX, which it saves in a local
129     ; variable and then returns to BDOS.
130
131     0056      Strat proc far
132     ; save address of Request Header
133     0056 2E: 89 1E 0012 R      mov      word ptr cs:[RH_Ptr],bx
134     005B 2E: 8C 06 0014 R      mov      word ptr cs:[RH_Ptr+2],es
135
136     0060 CB      ret      ; back to BDOS
137
138     0061      Strat endp
139
140     page
141
142     ; Device Driver "Interrupt Routine"
143
144     ; This entry point is called by the BDOS immediately after
145     ; the call to the "Strategy Routine", which saved the long
146     ; address of the Request Header in the local variable 'RH_Ptr'.
147
148     ; The "Interrupt Routine" uses the Command Code passed in
149     ; the Request Header to transfer to the appropriate device
150     ; handling routine. Each command code routine is responsible
151     ; for any necessary return information into the Request Header,
152     ; then transfers to Error or Exit to set the Return Status code.
153
154     0061      Intr proc far

```



```

156
157      0061 50      push    ax      ; save general registers
158      0062 53      push    bx
159      0063 51      push    cx
160      0064 52      push    dx
161      0065 1E      push    ds
162      0066 06      push    es
163      0067 57      push    di
164      0068 56      push    si
165      0069 55      push    bp
166
167      006A 0E      push    cs      ; make local data addressable
168      006B 1F      pop      ds
169
170      006C C4 3E 0012 R      les    di,[RH_Ptr]      ; ES:DI = Request Header
171
172                                ; get BX = Command Code
173      0070 26: 8A 5D 02      mov    bx,es:[di.Command]
174      0074 32 FF      xor     bh,bh
175      0076 83 FB 0F      cmp     bx,Max_Cmd      ; make sure its legal
176      0079 7F 06      jg      Unk_Command      ; too big, exit with error code
177      007B D1 E3      shl     bx,1      ; form index to Dispatch table
178                                ; and branch to driver routine
179      007D FF A7 0036 R      jmp     word ptr [bx+Dispatch]
180
181      page
182
183
184      ; General collection of exit points for the driver routines.
185
186
187      0081      Unk_Command:      ; Come here if Command Code too big.
188      0081 B0 03      mov     al,3      ; Sets "Unknown Command" error
189                                ; code and "Done" bit.
190
191      0083      Error:      ; Transfer here with AL = error code.
192      0083 B4 81      mov     ah,81h      ; Sets "Error" and "Done" bits.
193      0085 EB 03 90      jmp     Exit
194
195      0088 B4 01      Done:      mov     ah,1      ; Come here if I/O complete and
196                                ; no error, sets "Done" bit only.
197
198
199      008A      Exit:      ; General purpose exit point.
200                                ; Transfer here with AX =
201                                ; Return Status word to be
202                                ; placed into Request Header.
203
204      008A 2E: C5 1E 0012 R      lds     bx,cs:[RH_Ptr]      ; set status
205      008F 89 47 03      mov     ds:[bx.Status],ax
206
207      0092 5D      pop     bp      ;restore general registers
208      0093 5E      pop     si
209      0094 5F      pop     di
210      0095 07      pop     es
211      0096 1F      pop     ds
212      0097 5A      pop     dx
213      0098 59      pop     cx
214      0099 5B      pop     bx
215      009A 58      pop     ax
216      009B CB      ret      ; back to BDOS
217
218      page
219
220      ;
221      ; Function 1 Media Check
222      ;
223      009C      Media_Chk:
224      009C EB EA      jmp     Done
225
226      ;
227      ; Function 2 Build BIOS Parameter Block
228      ;
229      009E      Build_Bpb:
230      009E EB E8      jmp     Done
231

```

(Continued on next page)

MYSTIC PASCAL IS 10 TO 1000 TIMES FASTER THAN TURBO

Mystic Pascal compiles at over 100,000 lines per minute on a standard IBM PC. How? When you change a few lines of code, other Pascals make you recompile the whole program—Mystic only recompiles those lines. As if that's not fast enough, the compiler even runs in the background while you are editing. You can recompile a 2000 line program usually in less than one second.

Mystic produces 8086 object code, optimized on two levels. The single precision floating point is 5 to 50 times faster than any other compiler. 4000 multiplications or 2000 divisions per second, without an 8087—compare that to your present Pascal!

Still not sold?! OK. Mystic Pascal is also interactive—Pascal statements can be instantly compiled and executed. And there's a Full Screen Editor. And Help windows for the Standard Pascal language. And support for multi-tasking. And its only—

\$39.95!

Requires an IBM PC or true compatible with 256K. Turbo Pascal is a registered trademark of Borland International, Inc.

MYSTIC CANYON SOFTWARE
P.O. Box 1010
Pecos, New Mexico 87552

Place your order today!
Phone or use the coupon!
(505) 988-4214

Name _____

Address _____

City _____

State _____ Zip _____

Price is \$39.95 plus \$4 shipping.
Outside US & Canada add \$20 shipping.
Payment must be in US funds on a US bank. Purchase orders accepted from recognized institutions.
NM residents add sales tax.

☐ Check/MO ☐ COD ☐ VISA ☐ MC

Card _____

Exp. _____

Signature _____

16-Bit Toolbox Listing

(Listing Continued, text begins on page 94)

232					
233					
234					
235	00A0				
236	00A0	EB	E6		
237					
238					
239					
240					
241	00A2				
242	00A2	EB	E4		
243					
244					
245					
246					
247	00A4				
248	00A4	EB	E2		
249					
250					
251					
252					
253	00A6				
254	00A6	EB	E0		
255					
256					
257					
258					
259	00A8				
260	00A8	EB	DE		
261					
262					
263					
264					
265	00AA				
266	00AA	EB	DC		
267					
268					
269					
270					
271	00AC				
272	00AC	EB	DA		
273					
274					
275					
276					
277	00AE				
278	00AE	EB	D8		
279					
280					
281					
282					
283	00B0				
284	00B0	EB	D6		
285					
286					
287					
288					
289	00B2				
290	00B2	EB	D4		
291					
292					
293					
294					
295	00B4				
296	00B4	EB	D2		
297					
298					
299					
300					
301	00B6				
302	00B6	EB	D0		
303					
304					

;					
;	Function 3	I/O Control Read			
;					
;	Ioctl_Inp:				
	jmp	Done			
;					
;	Function 4	Read from Device			
;					
;	Input:				
	jmp	Done			
;					
;	Function 5	Non-destructive Read			
;					
;	Nd_Input:				
	jmp	done			
;					
;	Function 6	Return Input Status			
;					
;	Inp_Stat:				
	jmp	Done			
;					
;	Function 7	Flush Input Buffers			
;					
;	Inp_Flush:				
	jmp	Done			
;					
;	Function 8	Write to Device			
;					
;	Output:				
	jmp	Done			
;					
;	Function 9	Write with Verify			
;					
;	Outp_Vfy:				
	jmp	Done			
;					
;	Function 10	Return Output Status			
;					
;	Outp_Stat:				
	jmp	Done			
;					
;	Function 11	Flush Output Buffers			
;					
;	Outp_Flush:				
	jmp	Done			
;					
;	Function 12	I/O Control Write			
;					
;	Ioctl_Outp:				
	jmp	Done			
;					
;	Function 13	Device Open (MS-DOS 3.x)			
;					
;	Dev_Open:				
	jmp	Done			
;					
;	Function 14	Device Close (MS-DOS 3.x)			
;					
;	Dev_Close:				
	jmp	Done			
;					

(Continued on page 108)

Circle no. 30 on reader service card.

HOW DO YOU MEASURE A C COMPILER?

• CODE SPEED & SIZE

The Lattice C Compiler "generates code that is quite compact and fast running." *Peter Norton, PC Magazine*

• CONSISTENT RELIABILITY

"The Lattice Compiler has performed reliably and predictably." *R. Phraner, Byte Magazine*

• THIRD-PARTY LIBRARIES

More than 40 library products are currently available from Lattice

• DEBUGGER SUPPORT

The Lattice C-SPRITE Debugger is now available

• ALL MEMORY MODELS

Lattice C has 7 memory models available to allow the best solution for the task at hand

PLUS:

- **COMPILE TIME**
- **UNIX V COMPATIBILITY**
- **DOCUMENTATION**
- **UPDATE POLICIES**
- **COOPERATING PRODUCTS**
- **AVAILABILITY OF CROSS COMPILERS**
- **VENDOR REPUTATION**

SATISFACTION GUARANTEED!

Ask About Our "Trade Up To Lattice C" Policy



LATTICE

Lattice®, Inc.
P. O. Box 3148
Glen Ellyn, IL 60138
(312) 858-7950
TWX 910-291-2190

International Sales Offices

Belgium: Softshop. Phone: (32) 53-664875.

England: Roundhill. Phone: (0672) 54675.

Japan: Lifeboat. Phone: (03) 293-4711.

Circle no. 58 on reader service card.

Chassis Classics

3315
5" Floppy/Winchester
7 Cards **\$417***
\$100

3310
5" Floppy/Winchester
4 Cards **\$387***
\$100

3002T
5" Floppy/Winchester
10 Cards **\$565***
\$100

3307
8" Floppy/5" Winchester
7 Cards **\$494***
\$100

laser 3000

MAIN/FRAMES & DISC ENCLOSURES FROM \$100

LASER 3000 DISC/COVERS (not shown)

* 1 piece; prices lower in quantity.

3916F 5" Floppy **\$100*** **3915** 2 ea. 5" Winchester **\$199***

INTEGRAND

RESEARCH CORPORATION

(Disk drives not included)

8620 Roosevelt Ave./Visalia, CA 93291 209/651-1203

Circle no. 15 on reader service card.



Variable 54, Where Are You?

Is programming the latest game of Trivial Pursuits?

- Is this the latest listing?
- Where else is this variable changed?
- Where is this procedure used?

TSF's Source Locator helps you stay productive

Source code listings

- volume, path, file name, and time-stamp
- your project and proprietary information markings
- line numbers, line-wrap at word boundaries, and more

Cross-reference listings (data and procedures)

- assembler, BASIC, C, and PASCAL
- usage (value or assignment) and scope (global or local)
- your comments

Combine any number of files and languages to provide a comprehensive index for an entire program or system.

The Source Locator

Introductory Price **\$29.95**

For the IBM PC, XT, AT and compatibles, DOS 2.0+

TSF Dept. A-1

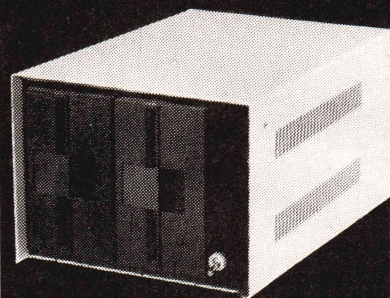
649 Mission St. San Francisco, CA. 94105

(415) 957-0111

Visa and Mastercard phone orders accepted. California residents add sales tax. Price includes shipping & handling. Dealer and Site License inquiries invited. Not copy protected.

Circle no. 34 on reader service card.

"Little Board" MAIN/FRAMES 6 Models from \$125*



\$150 (1 piece*)

MODEL 2800

Includes power supply & fan

(Disk Drives and Little Board not included)

AMPRO & Little Board are TM AMPRO computers.

INTEGRAND

RESEARCH CORPORATION

8620 Roosevelt Ave./Visalia, CA 93291

209/651-1203


```

305 ; Function 15 Removeable Media (MS-DOS 3.x)
306 ;
307 00B8 Rem_Media:
308 00B8 EB CE jmp Done
309
310 page
311
312 ; The Initialization code for the driver is called only once
313 ; when the driver is loaded. In this example, it returns its
314 ; own address to the DOS as the start of free memory after the
315 ; driver, so that the memory occupied by INIT will be reclaimed
316 ; after it is finished with its work. Only MS-DOS services 01-0CH
317 ; and 30H can be called by the INIT code.
318 ;
319 ; Block device drivers must also return the number of units and
320 ; the address of the BIOS Parameter Block pointer array; if all
321 ; units are the same, all pointers can point to the same BPB.
322 ;
323
324 00BA Init: ; Function 0
325 ; initialize device driver
326 00BA 06 push es ; push Request Header addr
327 00BB 57 push di
328 00BC B4 09 mov ah,9 ; print sign-on message
329 00BE BA 0016 R mov dx,offset Ident
330 00C1 CD 21 int 21h
331 00C3 5F pop di ; restore Request Header addr
332 00C4 07 pop es
333 ; set first usable memory addr.
334 00C5 26: C7 45 0E 00BA R mov word ptr es:[di.Address],offset Init
335 00CB 26: 8C 4D 10 mov word ptr es:[di.Address+2],cs
336 00CF EB B7 jmp Done
337
338
339 00D1 Intr endp
340
341 00D1 Driver endp
342
343 00D1 code ends
344
345 end

```

Structures and records:

Name	Width Shift	# fields Width Mask	Initial
REQUEST.	0016	0009	
RLENGTH.	0000		
UNIT.	0001		
COMMAND.	0002		
STATUS.	0003		
RESERVE.	0005		
MEDIA.	000D		
ADDRESS.	000E		
COUNT.	0012		
SECTOR.	0014		

Segments and Groups:

Name	Size	Align	Combine Class
CODE	00D1	PARA	PUBLIC 'CODE'

Symbols:

Name	Type	Value	Attr
BUILD_BPB.	L NEAR	009E	CODE
CR.	Number	000D	
DEV_CLOSE.	L NEAR	00B6	CODE
DEV_OPEN.	L NEAR	00B4	CODE
DISPATCH.	L NEAR	0036	CODE
DONE.	L NEAR	0088	CODE


```

DRIVER . . . . .
EOM. . . . .
ERROR. . . . .
EXIT . . . . .
HEADER . . . . .
IDENT. . . . .
INIT . . . . .
INPUT. . . . .
INP_FLUSH. . . . .
INP_STAT . . . . .
INTR . . . . .
IOCTL_INP. . . . .
IOCTL_OUTP . . . . .
LF . . . . .
MAX_CMD. . . . .
MEDIA_CHK. . . . .
ND_INPUT . . . . .
OUTPUT . . . . .
OUTP_FLUSH . . . . .
OUTP_STAT. . . . .
OUTP_VFY . . . . .
REM_MEDIA. . . . .
RH_PTR . . . . .
STRAT. . . . .
UNK_COMMAND. . . . .

```

```

F PROC 0000 CODE Length =00D1
Number 0024
L NEAR 0083 CODE
L NEAR 008A CODE
L DWORD 0000 CODE
L BYTE 0016 CODE
L NEAR 008A CODE
L NEAR 00A2 CODE
L NEAR 00A8 CODE
L NEAR 00A6 CODE
F PROC 0061 CODE Length =0070
L NEAR 00A0 CODE
L NEAR 00B2 CODE
Number 000A
Number 000F
L NEAR 009C CODE
L NEAR 00A4 CODE
L NEAR 00AA CODE
L NEAR 00B0 CODE
L NEAR 00AE CODE
L NEAR 00AC CODE
L NEAR 00B8 CODE
L DWORD 0012 CODE
F PROC 0056 CODE Length =000B
L NEAR 0081 CODE

```

49190 Bytes free

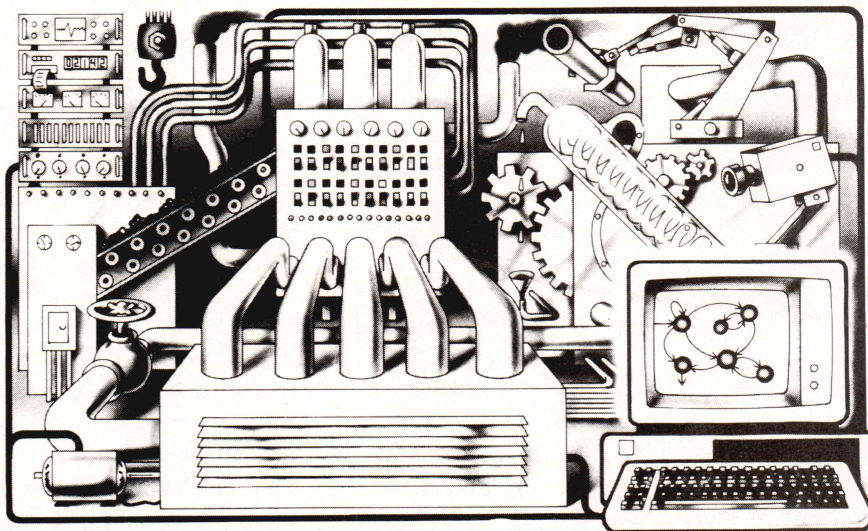
```

Warning Severe
Errors Errors
0 0

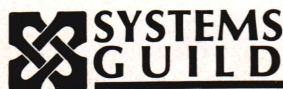
```

End Listing

Csharp Realtime Toolkit



Realtime on MSDOS? Csharp can do it! Get the tools without operating system overhead. Cut development time with C source code for realtime data acquisition and control. **Csharp** includes: graphics, event handling, procedure scheduling, state system control, and interrupt handling. Processor, device, and operating system independent. **Csharp** runs standalone or with: MSDOS, PCDOS, or RT11. **Csharp** runs on: PDP-11 and IBM PC. **Csharp** includes drivers for Hercules and IBM graphics boards, Data Translation and Metrabyte IO boards, real time clock, and more. Inquire for Victor 9000, Unix, and other systems. Price: \$600



Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

Circle no. 50 on reader service card.

by Michael Wiesenber

I've received a lot of response to the first two "Computer Calisthenics" columns.

The first puzzle in the November 1984 issue was:

There are two world-famed mathematicians, Mr. P and Mr. S. A friend of theirs who likes puzzles tells both of them that he has two numbers in mind. The only thing he will tell both of them about these two numbers is that they are both integers, greater than 1, and they are not the same.

He then whispers to Mr. P the product of the two numbers.

He whispers to Mr. S the sum of the two numbers.

Mr. P knows that Mr. S knows the sum, but he doesn't know what that sum is. Similarly, Mr. S knows that Mr. P knows the product, but he doesn't know what that product is.

The following conversation then takes place:

Mr. P: "I don't know the numbers."

Mr. S: "I know that; I also don't know the numbers."

Mr. P: "Oh, then I know the numbers."

Mr. S: "Really, then I do also."

Your task, of course, is to discover the two numbers. It would take you a long time to figure them out by hand, so write a program to do it. A short program.

Many offered programmatic solutions that produced the correct answer, but, unfortunately, they also yielded other sets of answers that were wrong.

Let's see what's happening at each statement. I'm going to give Alan Tracht, of Cleveland Heights, OH, an honorable mention for coming up with the correct reasoning required to solve the puzzle programmatically,

but he doesn't get the t-shirt because his program is one of those that yields too many answers:

(1) Mr. P: "I don't know the numbers." They are not both prime.

(2) Mr. S: "I know that . . ." The numbers cannot be the sum of two primes.

(3) ". . . I also don't know the numbers." The sum is at least 7, so that it can be the sum of at least two different pairs of numbers.

(4) Mr. P: "Oh, then I know the numbers." Of all the factor-pairs of the product, only one sums to a number that cannot be the sum of two primes.

(5) Mr. S: "Really, then I do also." Of all the addend-pairs, only one multiplies to a product that gives Mr. P the answer (under the conditions of the previous statement).

Let's see why certain sets of numbers fail one or more of the tests.

(5, 7): $P=35$, $S=12$

Mr. P would not have made statement 1.

(5, 6): $P=30$, $S=11$

Knowing the product, Mr. P would have made his first statement. Knowing the sum, Mr. S would have made statements 2 and 3. At this point, however, Mr. P still would not know whether his product of 30 was formed from the pair (5, 6) or (2, 15), and so would not have made statement 4.

(4, 61): $P=244$, $S=65$

This was the most common incorrect answer. These two numbers do not satisfy statement 5, because (4, 61) and (8, 57) each satisfy statement 4.

Mike Meyer, of Norman, OK, provided an interesting extension of this

problem, showing how this puzzle is the first of an infinite sequence of puzzles; that is, the conversation becomes:

Mr. P: "I don't know the numbers."

Mr. S: "I know that; I also don't know the numbers."

Puzzle 1:

Mr. P: "I still don't know the numbers."

Mr. S: "I still don't know the numbers, either."

Puzzle 2:

and so on. At each point, which we can label n , insert the sequence:

Mr. P: "Oh, then I know the numbers."

Mr. S: "Really, then I do also."

to create puzzle number n .

Interesting. The question is: who can solve one of those now?

The award goes to Charles Wells of the Department of Mathematics and Statistics, Case Western University, Cleveland, OH. The answer spit out by his program (Listing One, page 114) after running all night is (4, 13). He ran the program up to number pairs including 40, at which point he says it ran very slowly. He also implies that there might be larger answers, but "I refuse to believe anyone, mathematician or otherwise, could work this out for numbers above 20 in his head." Agreed. Perhaps I should have added to the puzzle the proviso that neither number was greater than 100, but nowhere did I say that the two world-famed mathematicians were idiot savants. Each would know that the other could not come up with an answer in his head for numbers greater than 100.

Although scores of entrants tried their hand at the preceding problem in logic, only two essayed the much easier task of programmatically describing a horse. J. C. Williams, of Cincinnati, OH, wins the prize for his program, presented in its entirety in these two lines:

```
FUNCTION HORSE( ) RETURN
  BOOLEAN
RETURN FALSE
```

He includes this "proof": "The function (program) is quite obviously a neigh-sayer," and appends the comment "(Sorry)." I presume he is apologizing for the pun, but that sort of originality was precisely what I was hoping for.

December's problem inspired more than double the entries of November's:

Assign a numerical value to each letter of the alphabet, starting with 1 for A and going up to 26 for Z. Any word in the English language has a value obtained by multiplying the values for each of its letters. For example, the word hello is worth 86,400, obtained by multiplying $8 \times 5 \times 12 \times 12 \times 15$. Which English word is equal to exactly 1,000,000? If there is none, which is the closest? Only words found in The Random House Dictionary of the English Language (unabridged edition) can be used. No capitalized words, none with hyphens or other embedded punctuation, nor those designated as foreign.

Well, how about it, folks? Can you devise a program that finds the right word? And, having done that, can you tell us what that word is? Your program must be short and elegant. The algorithms can be demonstrated in a good pseudolanguage if you wish or perhaps in flowcharts.

Several entrants sent me words that multiplied out to 1,000,000, but I did not award any of them a t-shirt for several reasons. Many discovered that the word *typey* has a value of 1,000,000 and told me what dictionaries they had found it in. Others offered *tetty*, defined in the *Oxford English Dictionary (OED)* as a form of "testy." (It's also in my *Webster's*

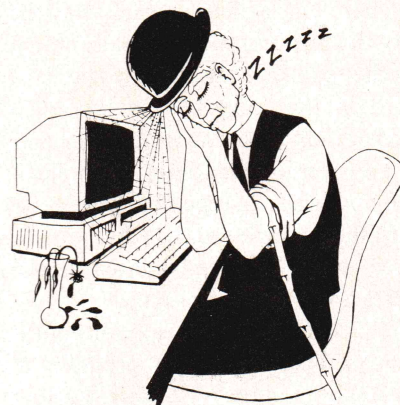
New International Dictionary, second edition.) The *OED* also yields *tytte*, an old form of "teat."

Those who sent me only one of these words did not qualify for the prize, because I asked for a *program*, not just an answer. Those who included a program that yielded only one of these words also did not win, because they did not follow the rules. I specified the *Random House Dictionary* for a number of reasons, and it does not contain *typey*, *tetty*, or *tytte*.

Why this particular dictionary? It is available as a dictionary program and thus does not necessitate typing in a whole dictionary full of words. Also, it is free of what I call "cross-word puzzle dictionary words"—obsolete words, archaic variants, and other words that are seldom found in use today. Furthermore, *not all the words in it are in the electronic dictionary*. No currently available electronic dictionary even comes close to having as many words as its "hard copy" counterpart does. It is an unusual spelling checker that has even 50,000 words, and yet the tome of over 2000 pages that is my main reference work contains in excess of 260,000 entries, and there are many more words than entries. For example, *tablet* is a separate entry, but *tableted* is not. *Kaleidoscopically* is not a separate entry; it is found at the end of the definition for *kaleidoscope*. Spelling checkers are even more limited: they usually have only one form of each word. The plurals of nouns, the progressive forms of verbs, and so on, are often not there. The point I wished to make, which most people did not get, was that perhaps brute force, in the form of examining every word in a given dictionary, is not the way to solve this puzzle.

I commend all those who wrote programs generating all possible candidates and embedded in their programs the logic that the letters forming the hypothetical word must be multiples of 2 or 5 or both, those being the only prime factors of 1,000,000; that is, if a word exists in the specified dictionary that multiplies out exactly to 1,000,000, it must consist of some combination of B, D, E, H, J, P, T, and Y. Any qualifying word can have,

BORED?...
...waiting
for **C** programs to
compile and link?



Use C-terp
the complete C interpreter

This is the product you've been waiting (and waiting) for!

Increase your productivity and avoid agonizing waits. Get instant feedback of your C programs for debugging and rapid prototyping. Then use your compiler for what it does best...compiling efficient code ...slowly.

C-terp Features

- Full K&R C (no compromises)
- Complete built-in screen editor--no half-way house, this editor has everything you need such as multi-files, inter-file move and copy, global searching, auto-indent, tab control, and much more.
- Fast--Linking and semi-compilation are breath-takingly fast. (From edit to run completion in a fraction of a second for small programs.)
- Convenient--Compiling and running are only a key-stroke or two away. Errors direct you back to the editor with the cursor set to the trouble spot.
- Object Module Support--Access functions and externals in object modules produced by C86 or Lattice C or assembly language. Utilize your existing libraries unchanged!
- Complete Multiple Module Support--Instant global searches, auto-compile everything that's changed, etc.
- Many more features including batch mode, 8087 support and symbolic debugging.
- Runs on IBM PC, DOS 2.x, 192K and up.
- **Price: \$300.00 (Demo \$45.00) MC, VISA**

Price of demo includes documentation and shipping within U.S. PA residents add 6% sales tax. Specify C86 or Lattice version.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: C86 (Computer Innovations), Lattice (Lattice Inc.), IBM (IBM Corp.), C-terp (Gimpel Software)

in addition, any number of As, that being a "free" letter because its addition to a word merely multiplies the product by 1. In fact, Ulrich Sondermann, of Concord, CA, chastised me for specifying the values of the letters to be 1 through 26, rather than 2 through 27—which, he said, was the way the problem was originally stated in the puzzle magazine. (The best word in that contest was *ixodid*, a kind of tick; Ulrich submitted the correct answer in that contest.) Ah, but I did not wish to repeat a previously published puzzle, and I *did* want a "free" letter because I wanted to provide a different kind of puzzle.

Ulrich deserves mention for another reason. He wrote his program in BASIC on a Timex/Sinclair: quite a feat! It generated 123 candidates (using the values I specified) in four minutes. He also provided a good algorithm for finding out if a given electronic dictionary contains any words consisting of these letter combinations. He said, "The problem then consists of creating a dictionary with words that have letters sorted in descending order. A match can then be made with each of the 123 candidates." (But what about the 124th candidate, Ulrich?)

I commend those of you who wrote such programs, but I did not award you prizes because I am not yet convinced that the *Random House Dictionary* has a word multiplying out to exactly 1,000,000 and because, with one notable exception, none of the programs accounted for the fact that adding any number of As to a word does not change its value.

Among the preceding, I give honorable mention to both Ken Waldron, of Vancouver, BC, and Ray Gardner, of Englewood, CO, because each devised a program that produced the same 124 candidates (all sans As). I also include Ray's program (Listing Two, page 116), because he accomplished in 24 lines of C code (seven lines of which, in good C coding practice, consist merely of one brace) what took Ken a page and a half of Pascal. Ray claims his program, written in Lattice C, runs in under four seconds on his IBM PC. Ray also says that his wife found the word *typey* in

15 minutes, without the benefit of a computer. (So much for computers.)

I also mention Thomas Shores, of Lincoln, NB, because, while his C program does not specifically generate words with As, he did acknowledge in his accompanying letter that they should be inserted "appropriately" within generated candidates. Also, his program generates all possible candidates, not just those multiplying out to 1,000,000. The best word he found was *moors*, value 1,000,350, but he thought others might come up with better possibilities. He stated that if any better words were found, they would have one of these values: 999,702; 999,810; 999,856; 1,000,000; 1,000,188. Good figuring, particularly since the word *I* think is the best has a value of 1,000,188. (How come you missed *rooms*, Thomas, an anagram of *moors*, and *comers*, which is also value 1,000,350?)

Trent Garverick, of Columbus, OH, gets an honorable mention for his short and sweet self-documenting pseudocode algorithm, the implementation of which in UCSD Pascal on an Apple II yielded what I believe to be the word *closest* to 1,000,000 found in the specified dictionary. *Curing* multiplies out to 1,000,188.

My own program had come up with that very word. My program first generated candidates within plus or minus 1000 of 1,000,000. I then "eyeballed" the list for real words. When I found several within 350, I narrowed the variance. Each time I found a better word, I started again with the smaller offset. My program, however, was inefficient, it took forever to run, and it generated literally thousands of possibilities, mostly because it spit out all permutations of each "word," not just words sorted in alphabetic descending order.

Trent based his program on examination of a real dictionary, rather than on generation of candidates. I still don't think that is the best method, but his program at least does not assume that a word multiplying out to exactly 1,000,000 must exist nor does it refuse to account for words with As in them.

I also mention Tom Balon, of Apa-

lachin, NY, and Bob Smith, of Windsor, NY, who jointly produced a Pascal program on a VAX 11-785 to examine a dictionary of 33,595 words, coming up with *curing*. They also found, tied at 1,000,188, *Nicaragua*, but that word is disqualified as being a proper noun. Their program, while relatively short, was not self-documenting. Joe Celko, of Los Angeles, also deserves mention for a short, relatively easy-to-follow C program that produces, in combination with a spelling checker, lots of possibilities, the best of which he concludes is *curing*.

And now for the winner. The envelope, Randy, please. Ah, yes, Fred Smith, of Stoneham, MA, wrote a short C program into which you input a list of words; the program then determines the value of each word. The

goatees	997500
soybean	997500
stogy	997500
suety	997500
alkaline	997920
budlike	997920
chuckle	997920
clinked	997920
driven	997920
flanker	997920
invader	997920
lurked	997920
pucker	997920
variant	997920
viaduct	997920
village	997920
mopped	998400
paldated	998400
banquet	999600
curing	1000188
nicaragua	1000188
comers	1000350
cozies	1000350
moors	1000350
rooms	1000350
seamier	1000350
arkansas	1000692
sulks	1000692
packets	1003200
shakeable	1003200
shelve	1003200
shoved	1003200
specked	1003200
stalked	1003200

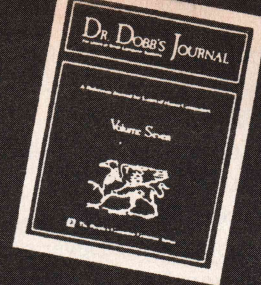
Figure
Partial Listing of Words
Produced by Fred Smith's Million.C

Announcing BOUND VOLUME 7

Every 1982 Issue Available For Your Personal Reference.

Vol. 7 1982

In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler, and we continued to publish utility programs and useful algorithms. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi supplied a year of "Clinic" columns while delivering his famous review of JRT Pascal and writing the first serious technical comparison of CP/M-86 and MSDOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."



Vol. 1 1976

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming which was then the only way to do things. This is always pertinent for bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing.

Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI, and more.

Vol. 2 1977

1977 found DDJ still on the forefront. These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 8080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NIBL.

Articles are about Lawrence Livermore Lab's BASIC, Alpha-Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities, and even more.

Vol. 3 1978

The microcomputer industry entered its adolescence in 1978. This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today.

Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors, and much, much more.

Vol. 4 1979

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages. Innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/Z80, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference.

Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe—more than ever!

Vol. 5 1980

All the ground-breaking issues from 1980 in one volume! Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a topic of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full! Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080, Further with Tiny BASIC, a Syntax-Oriented Compiler Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler and, as always, even more!

Vol. 6 1981

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features.

Highlights: information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

YES! ☐ Please send me the following Volumes of **Dr. Dobb's Journal**.
☐ ALL 7 for ONLY \$165, a savings of over 15%!

Payment must accompany your order.

Please charge my: ☐ Visa ☐ MasterCard ☐ American Express
I enclose ☐ Check/money order

Card # _____ Expiration Date _____

Signature _____

Name _____ Address _____

City _____ State _____ Zip _____

Mail to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303
Allow 6-9 weeks for delivery.

Vol. 1	x	\$26.75	=	_____
Vol. 2	x	\$27.75	=	_____
Vol. 3	x	\$27.75	=	_____
Vol. 4	x	\$27.75	=	_____
Vol. 5	x	\$27.75	=	_____
Vol. 6	x	\$27.75	=	_____
Vol. 7	x	\$30.75	=	_____
All 7	x	\$165.00	=	_____

Sub-total \$ _____

Postage & Handling _____
Must be included with order.
Please add \$1.25 per book in U.S.
(\$3.25 each surface mail
outside U.S. Foreign Airmail
rates available on request.)

TOTAL \$ _____

default is to print only those words whose values lie within 10,000 of 1,000,000, but it has a command line switch (great for debugging) that prints all values. The program also ignores words of less than five letters. And here is the clever part, the spark of originality for which Fred really deserves the t-shirt: he then used the powerful capabilities of Unix to examine an entire dictionary, one of over 90,000 words.

He used two Unix tools, each performing one task well. (One was his own word value-determination program.) His program, *million*, reads all the words in a 90,000-word dictionary called *words* and pipes its output to the *sort* utility, which performs a numeric sort on the second field of each line and writes the final sorted file to an output file called *m.w.sorted*. Fred's program is Listing Three

(page 18). Here is the Unix command:

```
million < words sort -n +1
> m.w.sorted
```

So why, if I'm not convinced that reading all the words of a dictionary—furthermore, one that does not contain all the possible words of the dictionary I specified—is the best method, do I award Fred the t-shirt? Two reasons. One, his program was the cleverest. Two, nobody did it the way I thought it should be done.

I was hoping that someone would devise a program that generates all candidates equal to and close to 1,000,000 and retains only those obviously or likely to be English words, coming up with a small list that a *human being* then could scan for "real" words. What I wanted to show was the necessary cooperation between

machines and humans on problems of this sort.

Russ Nelson, of Potsdam, NY, also deserves mention because he was clever enough to figure out precisely at which sprawling Silicon Valley corporate octopus my curious collection of I-Q Industries eccentrics *actually* works, proving to me that his knowledge was no mere guess by replicating the convoluted programmatic word game with which I produced the name. I would have awarded him an honorary t-shirt if he had also figured out what classical bit of entertainment inspired the transposition.

Thanks for all the kind letters!

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 197.

Computer Calisthenics (Text begins on page 110)

Listing One

```
PROGRAM DAVIS(INPUT,OUTPUT);
CONST LARGE=50;
VAR M,N:INTEGER;

(*AT EACH STAGE BELOW, N REFERS TO THE
NUMBER THAT MATHEMATICIAN WAS TOLD.*)
```

```
FUNCTION STAGEA(N:INTEGER):BOOLEAN;
```

```
(*AT STAGE A, MR. P DOESN'T KNOW.
THAT MEANS N HAS MORE THAN ONE
ACCEPTABLE FACTORIZATION.*)
```

```
VAR COUNT,F,G:INTEGER;
```

```
BEGIN
  F:=2; G:=N DIV 2; COUNT:=0;
  WHILE F<G DO
    BEGIN
      IF N MOD F = 0 THEN
        COUNT:=COUNT+1;
        F:=F+1; G:=N DIV F
      END;
    STAGEA:=(COUNT>1)
  END;
```

```
FUNCTION STAGEB(N:INTEGER):BOOLEAN;
```

```
(*AT STAGE B, MR. S KNOWS MR. P. DOESN'T
KNOW. THAT MEANS EACH ACCEPTABLE PARTITION
S+T=N YIELDS A NUMBER S*T WHICH SATISFIES
STAGE A. HE ALSO DOES NOT KNOW HIMSELF,
```

```
WHICH MEANS N HAS MORE THAN ONE ACCEPTABLE
PARTITION, WHICH MERELY MEANS N>6.*)
```

```
VAR S,T,COUNT:INTEGER;
TEMP:BOOLEAN;
```

```
BEGIN
  S:=2; T:=N-S; COUNT:=0; TEMP:=(N>6);
  WHILE S<T DO
    BEGIN
      COUNT:=COUNT+1;
      TEMP:=TEMP AND STAGEA(S*T);
      S:=S+1; T:=N-S
    END;
  STAGEB:=(COUNT>1) AND TEMP
END;
```

```
FUNCTION STAGEC(N:INTEGER):BOOLEAN;
```

```
VAR F,G,COUNT:INTEGER;
```

```
(*AT STAGE C, P KNOWS. THIS MEANS THERE
IS EXACTLY ONE ACCEPTABLE FACTORIZATION
OF M*N FOR WHICH STAGE B IS CORRECT*)
```

```
BEGIN
  F:=2; G:=N DIV F; COUNT:=0;
  WHILE F<G DO
    BEGIN
      IF (N MOD F = 0) AND STAGEB(F+G)
      THEN COUNT:=COUNT+1;
        F:=F+1; G:=N DIV F
      END;
    STAGEC:=(COUNT=1)
  END;
```

(Continued on page 116)

DDJ Classifieds

Software

DriveLiner

Drive Alignment Test Program for CP/M 2.2/3.1 With Dysan 8" SSD Diagnostic Disk \$65 check/MO postpaid Chandler Software, 273 W Shore Dr, Marblehead MA 01945 (617) 631-4685

NATIONAL PUBLIC DOMAIN

Why reinvent the wheel?

Public Domain Software isn't copyrighted, so no need to pay license fees for thousands of routines; business-utilities-dbase. Complete, latest issue user group disks available to rent 'n copy at your leisure. Send \$5.00 for a postpaid directory disk or a SASE to National Public Domain Software, 1533 Avohill Dr., Vista, CA 92083 or (619) 727-1015.

TECMAR GRAPHICS LIBRARY

TECMAR lets you do high-res graphics on your TECMAR Graphics Master. Features windowing, viewporting, clipping, axis rotation. Similar to Tektronix graphics. Includes screen dump/restore, Epson screen print, support for HP and Western Graphtec plotters. Includes three curve-fitting programs and graphics application SOURCE CODE. Requires MS-FORT 3.20, or Lahey F77L. Price: \$195. ADVANCED SYSTEMS CONSULTANTS, 18653 Ventura Boulevard, Suite 351, Tarzana, California 91356 (818) 990-4942

FED "Binary File Editor"

Allows the user direct access in both HEX and ASCII format to any disk file on the IBM PC. FED can patch object modules, examine word processing files, repair damaged files and verify the results of I/O operations. S/S DOS disk for 128 IBM PC. Only \$49 "+ \$5 s/h".

The Whitewater Group
2912 N. Burling Avenue
Chicago, IL 60657
(312) 975-6095

W.B. SOFTWARE DEVELOPMENT

BROWSE for CP/M-68K

for looking at any CP/M-68K file scroll up, down, left, right powerful FIND (search) command online HELP, PFKEYS, TABS, PRINT 100% 68000 assembler—ASCII, HEX ANSI 3.64 & other terminal types \$49.95 US. Visa, M/C, money order. WB Software Development, 112-Oakhampton Pl. SW, Calgary, AB, Canada T2V 4B2 (403) 238-3216

ECLIPSE SYSTEMS

AZTEC C65 + ProDOS

Use Aztec C DOS 3.3 software under ProDOS. Run programs without relinking. System includes recursive Shell, support for pseudo-code, vi like editor with macros, library upgrade and source code. Requires ProDOS user's disk. \$49.95. Eclipse Systems, 223 Matthew Rd., Marion, PA 19066. (215) 667-8354

Utility

MICRO COMPUTING SERVICES

CBTREE FOR C PROGRAMMERS

Provides enhanced file handling calls directly into C programs. Maintains balanced B-trees, supports unlimited number of keys and key lengths. Fast, Flexible, Efficient. No royalties. Source Code Incl. \$179. MICRO COMPUTING SERVICES 2009 Hileman Road Falls Church, VA 22043 (703) 893-0118

Publishing

OWINGS MILLS

OHIO SCIENTIFIC/ISOTRON AND COMPATIBLE COMPUTERS! Users MUST read PEEK (65). Published monthly exclusively for above users. \$19/yr U.S. \$26/yr Canada; P.O. Box 347, Owings Mills, MD 21117; (301) 363-3268.

SOFTWARE SENTINEL

A hardware key that prohibits unauthorized use. Its benefits: Unlimited backup copies; unbreakable; site licensing control; no floppy required with hard disk; transportable; transparent; pocketsize. Evaluation Kit available. PC compatible. **Rainbow Technologies, Inc.** 17971 Skypark Circle, Suite E Irvine, CA 92714 (714) 261-0228

DIAL & ORDER

24 HR. MODEM LINE

(408) 425-1371

Specializing in technical/reference books and computer supplies.

Wise Dog Computerbooks

1310 Fair Ave, Santa Cruz, CA 95060

Toll Free N. Cal. (800) 558-9473

CONTROL SYSTEMS SOFTWARE ENGINEER

A Challenging Opportunity in our Florida Corporate Offices

RS&H offers a challenging position with responsibility for development of systems level and applications level software for various Industrial Process Control applications.

Responsibilities will include involvement in associated hardware, and control strategy design, but a strong systems software background is the primary requirement.

Candidates should have at least three years' experience and a strong working knowledge of at least one computer operating system, assembly language, and "C". Familiarity with various DEC equipment operating systems, "C" and MACRO is particularly desirable. Experience with development of real-time software with some Process Control/Instrumentation background is desired. A BS degree in Computer Science or Engineering is a minimum requirement.

RS&H is a multi-disciplined A/E/P firm with a 43 year reputation for innovative technology applications and professional development. RS&H offers an attractive salary and benefits package including a prime Florida location providing a quality of life second to none. Please send your resume in complete confidence, including salary history to **Thomas M. Hills (904) 739-2000.**



REYNOLDS, SMITH AND HILLS

Architects-Engineers-Planners, Incorporated
P.O. Box 4850
Jacksonville, FL 32201

RS&H encourages qualified minorities, women, veterans and handicapped individuals to apply

EOE/AAP

Dr. Dobb's Journal is pleased to announce the DDJ Classifieds

RATES: DISPLAY ADVERTISERS: Price per column inch \$100. Ad must run in 3 consecutive issues.

LINE ADVERTISERS: Price per line \$12 (35 characters per line including spaces). Minimum of 5 lines. 3 consecutive issues. Add \$3 per line for boldface type. Add \$25 if a background screen is desired.

DISCOUNTS: Frequency discounts for 6 consecutive ads (less 7%) and for 12 consecutive ads (less 15%).

MECHANICAL REQUIREMENTS: Camera ready art or typewritten copy only (phone orders excepted). Display: Specify desired size, include \$20 if reduction is required. Line: Specify characters in boldface, caps. Allow 6 weeks for publication.

PAYMENTS: Full payment in advance. Check, money order, Visa, M/C, AmEx.

Run this ad in _____ issues

Size: _____ col x _____ inches or 1 col x _____ lines

Payment by: _____ check _____ m/o _____ Visa _____ M/C _____ AmEx

Card # _____ Exp Date _____

Signature _____

Print Name _____

Company _____

Address _____

City _____ St _____ Zip _____

Phone _____ Current Subscriber _____

Mail to or phone Tim Ortiz, DDJ Classifieds, 2464 Embarcadero Way, Palo Alto, CA 94303 (415) 424-0600

Listing One

```
FUNCTION STAGED(N:INTEGER):BOOLEAN;
```

```
VAR S,T,COUNT:INTEGER;
```

```
(*AT STAGE D, S KNOWS TOO. THIS MEANS  
THAT THERE IS EXACTLY ONE ACCEPTABLE  
PARTITION S+T OF N FOR WHICH STAGE  
C IS CORRECT FOR S*T.*)
```

```
BEGIN
```

```
  S:=2; T:=N-S; COUNT:=0;
```

```
  WHILE S<T DO
```

```
    BEGIN
```

```
      IF STAGED(S*T) THEN COUNT:=COUNT+1;
```

```
      S:=S+1; T:=N-S
```

```
    END;
```

```
    STAGED:=(COUNT=1)
```

```
  END;
```

```
BEGIN (*MAIN*)
```

```
FOR M:=3 TO LARGE DO
```

```
  BEGIN
```

```
    WRITE(M:3);
```

```
    IF (M MOD 12 = 0) THEN WRITELN;
```

```
    FOR N:= 2 TO M-1 DO
```

```
      IF STAGEA(M*N) THEN
```

```
        IF STAGEB(M+N) THEN
```

```
          IF STAGED(M*N) THEN
```

```
            IF STAGED(M+N) THEN
```

```
              BEGIN
```

```
                WRITELN;
```

```
                WRITELN('THIS PAIR WORKS:',N:3,M:3);
```

```
              END;
```

```
            END;
```

```
          END.
```

End Listing One

Listing Two

```
/*      solve  --  Computer Calisthenics solution  
*/
```

```
char letter [] = "bdehjpty";  
char solution[20];
```

```
main ()
```

```
{
```

```
  solve(0,0,1000000L);
```

```
}
```

```
solve (length, nextletter, goal)
```

```
int length, nextletter;
```

```
long goal;
```

```
{
```

```
  int i, value;
```

```
  if ( goal == 1 ) { /* found a solution, write it */
```

```
    solution[length] = '\0';
```

```
    puts(solution);
```

```
  } else { /* partial solution, extend it */
```

```
    for ( i = nextletter; letter[i]; i++ ) {
```

```
      value = letter[i] - 'a' + 1;
```

```
      if ( goal % value == 0 ) {
```

```
        solution[length] = letter[i];
```

```
        solve(length+1,i,goal/value);
```

```
      }
```

```
    }
```

```
  }
```

```
}
```

End Listing Two

(Continued on page 118)

DSD 80

FULL SCREEN SYMBOLIC DEBUGGER

**"THE SINGLE BEST DEBUGGER
FOR CP/M-80. A TRULY
AMAZING PRODUCT."**

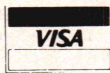
LEOR ZOLMAN
AUTHOR OF BDS C

- Complete upward compatibility with DDT
- Simultaneous instruction, register, stack & memory displays
- Software In-Circuit-Emulator provides write protected memory, execute only code and stack protection.
- Full Z80 support with Intel or Zilog Mnemonics
- Thirty day money back guarantee
- On-line help & 50 page user manual

**NOW
ONLY \$125.**

SOFTADVANCES

P.O. BOX 49473 AUSTIN, TEXAS 78765 (512) 478-4763



AT LAST

S-100 ↔ 488

THAT

DOES

EVERYTHING

YOU WANT

IT TO DO

Circle no. 63 on reader service card.

A FULL C COMPILER \$49⁹⁵ FOR

The Ecosoft Eco-C88 compiler for the 8088 and MSDOS is going to set a new standard for price and performance. Consider the evidence:

Compiler	Eco-C88	Lattice (1)	C86 (1)
Seive	13	11	13
Fib	44	58	46
Deref	13	13	-
Matrix	21	29	27
Price	\$49.95	\$500.00	\$395.00

(1) Computer Language, Feb., 1985, pp.73-102. Reprinted by permission.

The Eco-C88 compiler is a full K&R C compiler that supports all data types and operators (except bit fields). Now look at the other features we offer:

- ★ 8087 co-processor support using a single library. If you install an 8087 later, the software will use it without having to recompile.
- ★ A robust standard library with over 150 functions, including transcendentals, color, and others.
- ★ OBJ output for linking with the MSDOS linker (LINK).
- ★ Error messages in English - no cryptic numbers to look up. A real plus especially if you're just getting started with C.
- ★ Easy-to-read and complete user's manual.
- ★ Works with all IBM and compatibles running MSDOS 2.0 (or later).
- ★ Plus many other features.

For \$10.00 more, we will include the source code for the C library functions (excluding transcendentals). For an additional \$15.00, we will include our ISAM file handler in OBJ format (as published in the **C Programmer's Library**, Que Publishing). The discount prices for the library source and ISAM only apply at the time the compiler is purchased. Please add \$4.00 to cover postage and handling. To order, call or write:

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220
(317) 255-6476



Eco-C (Ecosoft), MSDOS (Microsoft), UNIX (Bell Labs), CP/M (Digital Research), Z80 (Zilog), 8086, 8087, 8088 (Intel).

Circle no. 35 on reader service card.



D&W DIGITAL, INC.
20655 Hathaway Avenue
Hayward, California 94541
(415) 887-5711

Circle no. 28 on reader service card.

Listing Three

```

1      c  /*
2      c  -----> million.c <-----
3      c
4      c  Usage: million [-nl] < infile > outfile
5      c
6      c  I/O is from stdio. By default, only words with products >= 990000 and
7      c  <= 1010000 are output. If -nl switch is used ('no limits'), the
8      c  limit test is suppressed, and all words >= 5 letters in length are
9      c  printed.
10     c
11     c  */
12
13
14
15
16     #include <stdio.h>
17     #include <ctype.h>
18
19     #define OPT      1000000          /* optimum value          */
20     #define MIN      OPT - 10000     /* minimum acceptable    */
21     #define MAX      OPT + 10000     /* maximum acceptable    */
22     #define BOOL     unsigned short
23     #define MAXLINE  80              /* longest permissible input string */
24     #define TRUE     1
25     #define FALSE    0
26
27     main(argc,argv)
28     {
29         int argc;
30         char *argv[];
31         {
32             char word[MAXLINE];      /* input word to be examined */
33             register char *pw;       /* register pointer thereto   */
34             register int wordlen;    /* length of string stored in word */
35             register unsigned long prod; /* working word value */
36             register BOOL print = TRUE; /* ok to print word if TRUE */
37             register BOOL limit = TRUE; /* assume MIN-MAX range limits */
38
39             if ((argc > 1) && !strcmp(argv[1], "-nl"))
40             {
41                 limit = FALSE;      /* suppress limit test for printing */
42             }
43             while ( gets(word) )    /* until end of input, get next word */
44             {
45                 if ((wordlen = strlen(word)) < 5) /* filter short words */
46                     continue;
47                 for (print = TRUE, prod = 1, pw = word ; *pw ; pw++)
48                 {
49                     /* examine for non-alpha characters,
50                      calculate product */
51                     if ( ! isalpha(*pw) )
52                     {
53                         print = FALSE;
54                         break;
55                     }
56                     prod *= (tolower(*pw) - 'a' + 1);
57                 }
58                 if (((prod < MIN) || (prod > MAX)) && limit)
59                 {
60                     print = FALSE; /* check limit conditions */
61                 }
62                 if (print)
63                 {
64                     printf("%s",word); /* is it ok to print? */
65                     for ( ; wordlen < 15 ; wordlen++) /* print word */
66                     {
67                         putchar(' '); /* pad with blanks */
68                     }
69                     printf("%u\n",prod); /* print word value */
70                 }
71             }
72         }
73     }

```

End Listings

\$49.95 FMT \$49.95

Text Formatting System

FMT provides most of the features of the high-priced Text Formatters at our inexpensive price -- and it's easier to use, too! Note the features below:

- Easily configured to your printer. Configuration files for 20+ printer models are provided or generate your own.
- FMT gets the most from your printers by taking advantage of their special features, including condensed, double width, enhanced, double print, italics, elite, letter quality, multiple fonts, etc.
- Multiple modes and combinations of modes can be used on the same line or even in the same word.
- FMT works with your favorite editor!
- FMT uses meaningful mnemonic commands in the style of SCRIPT or ROFF (each command appears on its own input line), including commands for the various printing modes.
- No embedded control codes - you don't have to remember those strange escape/control sequences.
- FMT runs at the maximum speed your printer allows for each printing mode - graphics mode is not required.
- Standard formatting features provided, including headers and footers, automatic page numbering, text justification, tabs for table generation, and embedded files up to TEN deep.
- FMT automatically builds Table of Contents, List of Figures, and three level alphabetized Index.
- Detailed 100+ page manual profusely illustrated with examples.
- Works equally well with IBM-PC, TI-PC, IBM clones and look-alikes (PC-DOS/MS-DOS 128k). Also works with CP/M 8080 and Z80 systems with 64k.
- \$49.95 plus \$2.00 shipping and handling.

Specify system.

VISA and Master Card Accepted
Dealer Inquiries Welcome

TINY TEK, INC.

Route 1, Box 795
Quinlan, Texas 75474
(214) 447-3025

Circle no. 39 on reader service card.

Periscope ... A Direct Hit!

"A great debugger ... If you've been frustrated by programs that don't behave the same for the debugger as when running alone, or that crash altogether, you should check out Periscope ..."

PC REPORT, Boston Computer Society

"It works, and works well!! In the first day of use I finished up two weeks of problems!! Really excellent!!!"

Peter Loats, Periscope User

Periscope's differences hit home! Its breakout switch gives you control, even when interrupts are disabled. Periscope's unique power helps you solve the difficult problems. Debug device-drivers, memory-resident, and non-DOS programs.

Great for straightening out confused C pointers! Using its breakpoint power, you can stop on reads and writes to ranges of memory. Stop on registers, words, and bytes, too.

It's tough. Periscope's 16K RAM board is write-protected. Runaway programs can't touch crucial debugger code! Examine the system, safely, at any time: Periscope saves the interrupt vectors and buffers it uses, then restores them when done. It uses ROM BIOS calls for functions except file access, so DOS can't clobber itself.

It gets you moving. Its commands are similar to Debug's, so you'll master it quickly. On-line help is there when you need it. You can define the windows you want; you can design easy-to-read memory displays. Periscope supports C, Assembler, BASIC, and Pascal. Comprehensive symbol support gives you a roadmap through memory tying back to your source.

It's risk-free. Periscope is backed up by a 30-day, money-back guarantee! The board is warranted for a year. Technical Support? You get the best there is ... direct from Brett Salter, Periscope's author!

It requires. An IBM PC, XT, AT, Compaq, or close compatible; PC-DOS, 64K RAM; a disk drive and an 80-column monitor. With two monitors, Periscope's screen is displayed on the monitor not in use. With one monitor, a keystroke switches screens.

It's power you can afford. At \$295, you can afford Periscope's professional power. The system includes the memory board, breakout switch, debugger software, manual, and quick reference card.

Order now! Call toll-free (800) 722-7006 to place your order or to get more information. MasterCard/VISA accepted.

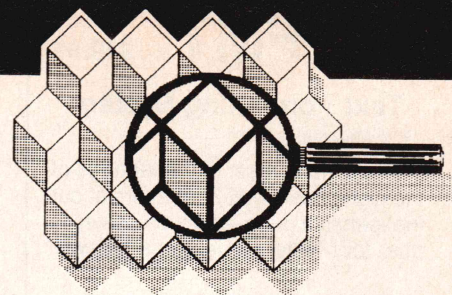
Get your programs up and running;

UP PERISCOPE!

Data Base Decisions/14 Bonnie Lane | Atlanta, GA 30328/(404) 256-3860



Circle no. 31 on reader service card.



by Michael Swaine

The marketing of microcomputer technology is not an exact science. Lee Felsenstein, designer of the Osborne 1 and a founder of the Community Memory Project that wrote the **Sequitur** relational database management system, is now selling Sequitur through his company, Goleemics, at a third of its original price. Sequitur was generally regarded as hot stuff when introduced some three years ago, but CM's faulty marketing kept it off the dealers' shelves, an article of furniture to which Sequitur will remain a stranger, since Sequitur will now be sold strictly by mail order at \$250. "I kept my mouth shut [earlier] because I thought I didn't know enough to criticize. My Osborne experience made me realize that nobody else knows any better either," Lee says. He's wiser now; he turned forty this spring . . . Micro Craft, manufacturer of the **Dimension** computer, a pricey multiprocessor machine curiously positioned on release as a do-all end-user product, has also recently risen from the rubble of bad marketing. The company is now selling dual-68000 Unix machines and enjoying post-Chapter 11 prosperity . . . New Functions, the distributor of **Judy**, a concurrent personal assistant (aka gal/man Friday, sidekick, right-hand man, desktop accessory or pop-up), has one of the more novel distribution schemes extant. Judy owners can generate and give away Judy "clones"—limited-time free samples—and when the sampler buys his own Judy, the clonemaker gets a commission. New Functions recently raised that commission from 10% to 20%.

The largest computer show in the world is not held in the U.S. Atari introduced its 68000-based **ST** computer at the mammoth Hannover Fair in Germany in April. Despite

the fact that the ST had a tendency to crash when asked to do anything much beyond the prepared demo, our correspondent on the scene found both the machine and its reception impressive, and advised us to look into software development for the ST under the GEM environment . . . Jack Tramiel's former company, Commodore, which has long stood strong in Europe, lost money this spring for the first time ever as dealers held out for the new **128** machine . . . A new **European Forth chip** could provide some competition for the **Novix NC4000 Forth chip**, although the Novix chip should actually be available by the time you read this. The NC4000 was designed by Charles Moore, who created Forth, and Robert Murphy; it can address four megabytes of memory and is intended to execute Forth code on the order of ten times the speed of any software implementation of Forth. Its designers expect it to find application in speed-critical areas like animation.

Boards

RAM price wars are behind price reductions on some hardware enhancements. The price of a do-it-yourself Macintosh upgrade is now below \$200 . . . DFE has cut prices on its **NS32032 coprocessor board** for IBM PC/XT/AT and PC-compatible computers. The basic Tiger-32 was introduced last summer with a 6MHz clock, 32082 MMU, 512K onboard RAM and Xenix 3.0 for \$2495, but RAM price-cutting has brought it down to \$2095 . . . Univation's **Turbocharger** is a 9.54 MHz 8086 board for the IBM PC that sells for \$1295 with 640K on-board RAM . . . Real Time Devices is selling a **GPiB instrumentation bus interface board** for

PCs for \$299, a per-channel cost of about \$20.

Some new boards refine existing capabilities: Interstate Voice Products announced a **speaker-dependent IBM PC board** that has some limited facility for recognizing words in connected speech . . . We're examining STB's **video board**, which runs all IBM-compatible software, whether it was designed for a color or a monochrome monitor, by converting graphics displays into full-screen 16-level grey-scale images . . . ANEX Technology is selling a **banked 2-megabyte RAM board** for PCs that are running multi-user software.

For \$61 (quantity one price), you can pick up a VLSI chip that lets you directly replace an 8088 microprocessor with an 80286. Edsun Labs' **EL286-88 Processor Converter** stands between the 286 and the peripherals, playing an 8088 to the peripherals, while accepting and converting all 16-bit transfer requests from the 286. The idea is to combine 16-bit main memory operations with low-cost 8-bit peripherals.

MSDOS

Alternatives or extensions to MSDOS/PCDOS are popping up like pop-ups. DRI has announced the first software to run under its Graphics Environment Manager (GEM): **GEM Desktop, Draw, Write, Paint, Wordchart and Graph**. Desktop lets you run up to six utility programs on top of applications, like the Mac's desktop accessories; all are supposed to be shipping as you read this . . . Digital's **pcShare** substitutes a multi-user system with file protection for PCDOS . . . RTCS's **PC/RTX** substitutes a real-time O/S (an implementation of Intel's iRMX-86) . . . Soft-

Logic's **DoubleDOS** makes PCDOS multitask for \$99 ("multi" here means "two").

For software development under MSDOS/PCDOS, Phoenix software has a program development manager called **Pmaker** that they say is similar to the Unix "make" command, and a program performance analyzer called **Pfinish** . . . **Advance Trace86** from Morgan Computing is a debugger/assembler that lets you back up execution 20 instructions . . . Genesis's **GeneScope** is a new full-screen symbolic debugger for the PC.

Whether developing expert systems on a micro is the Next Big Thing or just the latest fad, it's getting cheaper and easier. **Insight 2** from Level Five Research lets you incorporate Pascal routines and dBase II files into a knowledge base developed on a PC or compatible . . . AI pioneer Donald Michie's **Expert Ease** package has undergone some changes of ownership and distribution (it's now distributed by Human Edge), the most visible consequence of which is a price cut from \$2000 to \$695.

C

"To boldly go where no C software distributor has gone before . . ." Phoenix calls **Pre-C**, its lint superset, the industry's first program analyzer for the C language designed to support MSDOS and PCDOS . . . Micro Software Developers says its **C Debugger** is the first true source debugger for C . . . Catalytics has released what it calls the first complete C interpreter, its **Safe C** . . . For Lattice C programmers who don't necessarily hanker to be first, or at least would like some company out there on the giddy edge, Bill Hunt, author of *The C Toolbox*, has started the **Lattice C User's Group**.

Whitesmiths has upgraded its 8086-family **C compiler** to support all 8086 memory models and for closer adherence to their best guess at the emerging ANSI standard . . . Toolworks has moved its **C/80 compiler** to MSDOS and is selling it for \$49.95 . . . Smith & Smith Associates has released **CrossRefC**, a C cross-reference/listing generator, for \$39.95 . . .

Vance Info Systems announced **C Lib**, a C function library of over 200 routines, for the MSDOS environment.

CP/M

CP/M users needn't feel left out of the pop-up fad: Poor Person Software's **Write-Hand-Man** (that's something like a sidekick, we suppose) includes a notebook, phonepad, desk calendar, file and directory viewing programs and a communications program; new functions can be added fairly easily. It sells for \$49.95 and runs under CP/M 2.2 on all CP/M machines.

Also for CP/M, **MB+ Tools** is

Minnow Bear Software's \$175 set of programmer productivity tools for Pascal MT+ . . . Soft Advances says that its **DSD80 full-screen symbolic debugger** is fully DDT-compatible and fully supports Z80 instructions using either extended Intel or Zilog mnemonics . . . Selfware's **Convert 3.1** lets you read, write and format disks, copy files and list directories in 105 CP/M formats on an MSDOS machine, and costs \$99.

Mac

Just now, despite such attractions as the novelty of a Smalltalklike Forth

Thinking of... 8087, High-Speed Math Support?

THINK C-86

OPTIMIZING C86 FROM COMPUTER INNOVATIONS

The 8087/80287 Math Co-processor chips provide a major leap in the computing capability of the IBM family of personal computers. The fast, accurate, math computational ability provided by these chips not only speeds typical applications but opens a range of applications once reserved exclusively for mini and mainframe systems. The key question is HOW CAN THE C PROGRAMMER TAKE ADVANTAGE OF THE 8087/80287's capabilities? The answer is C86 by Computer Innovations.

EXECUTION SPEED, ACCURACY, IN-LINE CODE
C86 has offered the highest level of 8087/80287 support since the chips became available. Specialized libraries make program development a snap. A math function library provides full trig, log and a wide variety of other functions.

But most important, C86 produces **IN-LINE** code. This means that math functions can be performed by the co-processor in immediate succession WITHOUT separate subroutine calls. This eliminates library call "overhead" and results in the highest execution speed attainable.

LIBRARY SOURCE PROVIDES CONFIDENCE - CONTROL
Library Source Code is INCLUDED. - It always has been with C86. This means you have total control over your development environment, and total control means better and faster code.

TECH SUPPORT - GET INTO 8087/80287 PAINLESSLY
Computer Innovations' experienced tech support team is available to get you through the rough spots and provide the assistance/advice/expertise required to get you into 8087/80287 quickly and efficiently. Plus you can take advantage of Computer Innovations' user newsletter, technical notes, and On-Line bulletin board. No other Compiler offers more.

CALL FOR FREE BULLETIN
To receive Computer Innovations' free bulletin entitled "How C86 Takes Advantage of Intel 8087/80827 Math Co-processors," or for more information about C86 Call:

800-922-0160



**COMPUTER
INNOVATIONS, INC.**

980 Shrewsbury Avenue,
Tinton Falls, NJ 07724 (201) 542-5920

C86 for all PC-MSDOS computers. \$395.00. Visa, Mastercard, personal check and corporate PO's accepted.

Circle no. 96 on reader service card.

DDJ BACK ISSUES

#73 Volume VII, Issue 11:

Wildcard UNIX Filenames—Tests for Pidgin—68000 Cross Assembler Listing, Part 2—Adding More BDOS Calls—The Perfect Hash—BASIC Memory Management—Benchmarks for CP/M-86 vs. MSDOS, and the 8087.

#78 Volume VIII, Issue 4:

RECLAIM Destroyed Directories—Binary Magic Numbers—8080 Fig-Forth Directory & File System—SAY" Forth Votrax Driver—TRS-80 8080 to Z80 Translator—Basic Disk I/O, Part II.

#79 Volume VIII, Issue 5:

The Augusta Compiler—A Fast Circle Routine—Enhancing the C Screen Editor—Shifts and Rotations on the Z80—The SCB, TSX, and TXS Instructions of the 6502 and 6800—MS-DOS vs. CP/M-86—Controlling MBASIC—The Buffered Keyboard—IBM PC Character Set Linker—Flip Utility for the IBM PC.

#80 Volume VIII, Issue 6:

Fast Divisibility Algorithms—B-Tree ISAM Concepts—CP/M BDOS and BIOS Calls for C—Serial Expansion in Forth—Fast Matrix Operations in Forth, Part I—Yes, You Can Trace Through BDOS—Julian Dates for Microcomputers—8088 Addressing Modes—8088 Line Generator—CP/M Plus.

#81 Volume VIII, Issue 7:

The Augusta Compiler, continued—RED: A Better Screen Editor, Part I—Anatomy of a Digital Vector and Curve Generator—Fast Matrix Operations in Forth, Part II—The AGGHHH Program—MBOOT Revisited—CP/M Plus Feedback—MS-DOS Rebuttal—68000 Tools—Sizing Memory on the IBM PC.

#82 Volume VIII, Issue 8:

Serial-to-Parallel: A Flexible Utility Box—McWORDER: A Tiny Text Editor—And Still More Fifth Generation Computers—Specialist Symbols and I/O Benchmarks for CP/M Plus—CP/M Plus Memory Management—Zero Length File Test—PAUSEIF, QUITIF, and now SKIPIF—ACTxx Cross Assemblers.

#83 Volume VIII, Issue 9:

FORTH ISSUE: Forth and the Motorola 68000—Nondeterministic Control Words in Forth—A68000 Forth Assembler—GO in Forth—Precompiled Forth Modules—Signed Integer Division—Some Forth Coding Standards—The Forth Sort.

#84 Volume VIII, Issue 10:

Unix to CP/M Floppy Disk File Conversion—A Small-C Help Facility—Attaching a Winchester Hard Disk to the S-100 Bus—Using Epson Bit-Plot Graphics—8086/88 Function Macros—Auto Disk Format Selection—CP/M Plus Device Tables.

#85 Volume VIII, Issue 11:

A Kernel for the MC68000—A DML Parser—Towards a More Writable Forth Syntax—Simple Graphics for Printer—Floating-Point Benchmarks.

#86 Volume VIII, Issue 12:

Faster Circles for Apples—Cursor Control for Dumb Terminals—Dysan's Digital Diagnostic Diskette—Interfacing a Hard Disk Within a CP/M Environment—The New MS-DOS EXEC Function.

#87 Volume IX, Issue 1:

A structured Preprocessor for MBASIC—A Simple Window Package—Forth to PC-DOS Interface—Sorted Diskette Directory Listing for the IBM PC—Emulate WordStar on TOPS-20—More on optimizing compilers—The PIP mystery device contest.

#88 Volume IX, Issue 2:

Telecommunications Issue: Micro To Mainframe Connection—Communications Protocols—Unix to Unix Network Utilities—VPC: A Virtual Personal Computer for Networks—PABX on the Personal Computer—BASIC Language Telecommunications Programming—U.S. Robotics S-100 Card Modem.

#89 Volume IX, Issue 3:

RSA: A Public Key Cryptography System, Part I—Introduction to PL/C: Programming Language for Compilers—Program Design Using Pseudocode—More on Binary Magic Numbers—How fast is CP/M Plus?—CP/M 2.2 BIOS Function: SELDSK—The results of the Floating-Point benchmark.

#90 Volume IX, Issue 4:

Optimizing Strings in C—Expert Systems and the Weather—RSA: A Public Key Cryptography System, Part II—Several items on CP/M Plus, CP/M 2.2 Compatibility—BDOS Function 10: Vastly improved—More on MS-DOS EXEC Function—Low-Level Input-Output in C.

#91 Volume IX, Issue 5:

Introduction to Modula-2 for Pascal Programmers—Converting Fig-Forth to Forth-83—Sixth Generation Computers—A New Library for Small-C—Solutions to Quirks in dBASE II.

#92 Volume IX, Issue 6:

CP/M on the Commodore 64—dBASE II Programming Techniques—First Chinese Forth: A Double-Headed Approach—cc-A Driver for a Small-C Programming System—A New Library for Small-C (Part II)—Comments on Sixth Generation Computers—Review of Turbo Pascal.

#95 Volume IX, Issue 9:

Forth Special Issue!—File Maintenance in Forth—Forth and the Fast Fourier Transform—Computing with Streams—A Forth Native-Code Cross Compiler for the MC68000—The FVG Standard Floating-Point Extension—CP/M Plus: Interbank Memory Moves Without DMA—Ways to make C more powerful and flexible.

#96 Volume IX, Issue 10:

More dBASE II Programming Techniques—Simple Calculations with Complex Numbers—GREP.C: A Unix-like Generalized Regular Expression Parser—An optimization scheme for compilers, MSDOS 2.0 Filters, Sizing RAM under MSDOS, Two programming systems illustrating Runge-Kutta integration.

#97 Volume IX, Issue 11:

Adding Primitive I/O Functions to muLISP—Program Monitor Package: Using Interrupts to Instrument Applications—CP/M 2.2 Goes PUBLIC—A Guide to Resources for the C Programmer—RESORT.

#98 Volume IX, Issue 12:

Varieties of Unix—Unix Device Drivers—A Unix Internals Bibliography—A File Browser Program—An Introduction to Parsing.

#100 Volume X, Issue 2:

Festschrift for Doctor Dobb—Fire in the Valley—Tiny Basic for the 68000—An Enhanced ADFGVX Cipher System—More dBASE Tips & Techniques.

#101 Volume X, Issue 3:

Programming in Logic—Tour of Prolog—Tax Advisor: A Prolog Program Analyzing Tax Issues.

#102 Volume X, Issue 4:

What We Can Learn about Human Factors Engineering from a Game-Playing Computer—Shortcut to SCISTAR: For Prowriter Users—fx80char: A Character Editor for the Epson FX 80—A Magic Mushroom for the Epson Alice—Let's Mouse Around

#103 Volume X, Issue 5:

Using Decision Variables in Graphics Primitives—Solid Shape Drawing on the Commodore 64—A Compiler Written in Prolog

#104 Volume X, Issue 6:

Information Age Issues: Modems: 2400 Bit/Sec and Beyond; C UART Controller; Christensen Protocols in C

TO ORDER:

Send \$3.50 per issue to Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

Please send me the issue(s) circled: **71 72 73 78 79**
80 81 82 83 84 85 86 87 88 89 90 91 92
95 96 97 98 100 101 102 103 104

I enclose \$ _____ (U.S. check or money order).

Outside the U.S., add \$.50 per issue.

I have read the postal instructions and understand that I will not receive my order unless I have sent the correct payment amount.

Please charge my: ☐ Visa ☐ M/C ☐ Amer. Exp.

Card No. _____ Exp. Date _____

Signature _____

Please provide a street address rather than a P.O. Box.

Name _____

Address _____

City _____

State _____ Zip _____

Availability on first come/first serve basis. Outside the U.S. add \$.50 per issue ordered. Price includes issue, handling, and shipment by second class or foreign surface mail. Within the U.S., please allow 6-9 weeks to process your order second class. For faster service within the U.S., we'll ship UPS if you add \$1.00 for 1-2 issues and \$.50 for each issue thereafter. We need a street address, not a P.O. Box. Airmail rates: To Canada add \$1.75 per magazine, all other foreign add \$3.00 per magazine.

extension (Kriya's **Neon**), some of the most interesting products for the Mac are to be found on user group disks. Not surprisingly, a high proportion of these public-domain or user-supported programs consists of communications software, like Dennis Brothers' **MacTEP** and Scott Watson's **Red Ryder**. You can pick more fruit if you have a ladder ... There are also a lot of fonts, desk accessories and graphics; the **IM Underground's** graphics consist of schematics for the machine ... Equally unsurprising, given the features of the Mac, is the number of attractively-designed Mac newsletters, including **MacTutor**, **The Mac Street Journal** and **MacBriefs**.

Apple][

Contrary to the impression given at its stockholders' meeting early this year, Apple makes another computer, the Apple bracket-bracket. Programmers are even continuing to write software for the bracket-bracket ... MicroSparc has released the ProDOS version of its **Ampersoft** utility package at \$49.95 for Applesoft programmers ... There is, of course, much existing Apple][software: **InCider** is selling disks of software from the magazine at \$21.47 ... **Pandora Software** has collected a library of 4000+ public-domain programs for the Apple][and is selling them at \$5 per disk (at twenty programs per disk, that's \$1000 for the library, but still).

A Visit to the Fab Lab

A blip of *deja vu* struck as I read in the March *IEEE Software* about Ivan Guzman de Rojas and his plan for doing natural-language translation via the peculiarly algebraic Andean Indian language Aymara. Yes, trivial recall was working perfectly: halfway down the Andean peak of press releases on my desk I found the announcement of the opening of the **Aymara Fab Lab** in Sunnyvale. A coincidence worthy of a drive down the peninsula.

Dean Norman, Director of the Aymara Fab Lab, greeted me at the door and began talking immediately

about Guzman. Norman explained that Guzman, discovering that Aymara, lacking irregular verbs and gender, was an unprecedentedly logical language (although its logic was not standard two-valued logic), had succeeded in codifying the algorithmic structure of its syntax. For the first time, someone had expressed a natural language in software. Wasn't Guzman, I asked, considering the application of his achievement in the design of translating machines, the notion being that computerized Aymara could serve as the bridge in a multilanguage translation system?

Right, Norman answered, although at AFL they were taking the process in a somewhat different direction. Did I recall the Sapir-Whorf hypothesis from linguistics? I did, more or less: that language delimited the thinkable thoughts, and thus our culture and perceptions. Under normal conditions, we see only those distinctions for which we have words. In cultures in which green is not a major linguistic division of the spectrum, it is also not a primary perceptual division.

Norman nodded. The principle can be applied to any language-processing system, natural or artificial. Curious as it might sound, the Aymara-speaking software would, to a certain extent, think like an Aymaran. With its multi-valued logic, it would make distinctions that would never occur to a New York stockbroker; with its lack of grammatical gender, it would fail to make distinctions the stockbroker would unconsciously make. And Aymaran is only the easiest language, not the only one, to which the principle can be applied. Employing Guzman's translation techniques, it would be possible to develop front-end packages that, with the proper filtering out of Aymaran values and perceptions, would embody pure upper-class British perceptions or ancient Greek thought processes. We could examine the way judges in ancient Sumeria examined evidence.

That, Norman explained, was what they were up to at AFL: just as the developers of expert systems were trying to capture the knowledge of selected individuals in software, AFL was trying to capture the style of thinking,

YOU NEED A GOOD LIBRARY



COMPLETE SOURCES NO ROYALTIES

COMPREHENSIVE C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."

— Dr. Dobb's Journal, July 1984

USEFUL "...can be used as an excellent learning tool for beginning C Programmers..."

— PC User's Group of Colorado, Jan. 1985

FLEXIBLE Most Compilers and all Memory Models supported.

RECOMMENDED "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time." — Microsystems, Oct. 1984

- **PACK 1: Building Blocks I** \$149
DOS, Keyboard, File, Printer, Video, Async
 - **PACK 2: Database** \$399
B-Tree, Virtual Memory, Lists, Variable Records
 - **PACK 3: Communications** \$149
Smartmodem™, Xon/Xoff, X-Modem, Modem-7
 - **PACK 4: Building Blocks II** \$149
Dates, Textwindows, Menus, Data Compression, Graphics
 - **PACK 5: Mathematics I** \$99
Log, Trig, Random, Std Deviation
 - **PACK 6: Utilities I** \$99
(EXE files)
Arc, Diff, Replace, Scan, Wipe
- Master Card/Visa, \$7 Shipping, Mass. Sales Tax 5%
ASK FOR FREE DEMO DISKETTE

NOVUM
ORGANUM
INC.

SOFTWARE
HORIZONS
INC.

165 Bedford St., Burlington, MA 01803
(617) 273-4711

Circle no. 90 on reader service card.

LISP FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE
OF ARTIFICIAL
INTELLIGENCE FOR
YOUR IBM PC.

■ DATA TYPES

Lists and Symbols
Unlimited Precision Integers
Floating Point Numbers
Character Strings
Multidimensional Arrays
Files
Machine Language Code

■ MEMORY MANAGEMENT

Full Memory Space Supported
Dynamic Allocation
Compacting Garbage Collector

■ FUNCTION TYPES

EXPR/FEXPR/MACRO
Machine Language Primitives
Over 190 Primitive Functions

■ IO SUPPORT

Multiple Display Windows
Cursor Control
All Function Keys Supported
Read and Splice Macros
Disk Files

■ POWERFUL ERROR RECOVERY

■ 8087 SUPPORT

■ COLOR GRAPHICS

■ LISP LIBRARY

Structured Programming Macros
Editor and Formatter
Package Support
Debugging Functions
.OBJ File Loader

■ RUNS UNDER PC-DOS 1.1 or 2.0

■ IQLISP

5¼" Diskette
and Manual _____ \$175.00
Manual Only _____ \$ 30.00

∫q Integral Quality

P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

the intellectual spirit of whole cultures. I mulled that over. Wouldn't there be a great advantage, I asked, in combining the two approaches, developing a system with a specifiable knowledge base and a specifiable style of thinking? Couldn't one develop, say, a machine with the knowledge of a high-energy physicist and the spirit of a 12th-century Mandarin? Or the knowledge of a modern statesman and the intellectual style of the first Continental Congress? But Norman suddenly looked uncomfortable and said that he couldn't discuss details of ongoing projects.

Reference Map

Goleemics, Inc.

2600 10th St.

Berkeley, CA 94710

(415) 486-8347

Reader Service No. 101.

Micro Craft Corp.

4747 Irving Blvd., Ste 241

Dallas, TX 75247

(214) 630-2562

Reader Service No. 103.

New Functions

P.O. Box BB

Staten Island, NY 10302

(718) 273-2668

Reader Service No. 105.

Atari

1196 Borregas

Sunnyvale, CA 94086

(408) 745-2000

Reader Service No. 107.

Dr. Alan Winfield

Metaforth Computer Systems Ltd.

Unit 2 B The Newlands Center

Inglemire Lane

Hull HU6 7TQ

England

Reader Service No. 109.

Novix

10590 North Tantau Ave.

Cupertino, CA 95014

(408) 996-9363

Reader Service No. 111.

DFE Electronic Data Systems

USA: 5820 Stoneridge Mall Rd.

Suite 115

Pleasanton, CA 94566

(415) 847-2024

Europe: Singerstrasse 3,

7513 Stutensee

West Germany

07244-9011

Reader Service No. 113.

Real Time Devices, Inc.

1930 Park Forest Ave.

P.O. Box 906

State College, PA 16804

(814) 234-8087

Reader Service No. 115.

Interstate Voice Products

1849 W. Sequoia Ave.

Orange, CA 92668

(714) 937-9010

Reader Service No. 117.

STB Systems, Inc.

601 N. Glenville, Ste 125

Richardson, TX 75081

(214) 234-8750

Reader Service No. 119.

Univation

1037 North Fair Oaks Ave.

Sunnyvale, CA 94089

(408) 745-0180

Reader Service No. 121.

ANEX Technology, Inc.

151 North Route 9W

Congers, NY 10920

(914) 268-2400

Reader Service No. 123.

Edsun Laboratories, Inc.

7 Sears Rd.

Wayland, MA 01778

(617) 358-5667

Reader Service No. 125.

Digital Research Inc.

Box DRI

Monterey, CA 93942

(800) 443-4200

Reader Service No. 127.

Digitrol Computers Inc.

440 Phillip St.

Waterloo, Ontario

Canada N2L 5R9

(519) 884-4541

Reader Service No. 129.

RTCS Corp.



The New Aztec C

Better Code With Less Effort

The new 3.2 Version of Manx Aztec C86-c produces faster and smaller code with less effort than any C development system available for PC-DOS and MS-DOS.

"A compiler that has many strengths ... extensions seem to be quite valuable for serious work"
(Aztec C86 2.2 review in Computer Languages 2/85)

Better Code ...

Manx Aztec C86-c produces code that executes up to 60 percent faster in up to 60 percent less space than code produced by other PC-DOS and MS-DOS compilers. In short, it generates the best code available.

An extract from the C benchmark comparison in the January, 1985 issue of Computer Languages is reproduced here. Aztec C86-c clearly generated the best code. Modifying the sieve benchmark to use register variables presents an even clearer picture. Aztec C86-c executes in 6.51 seconds, Mark Williams executes in 7.56 seconds, and there is no improvement for Lattice and Computer Innovations Optimized C86. The Dhrystone benchmark results presented here are from a benchmark study conducted by MANX. The Dhrystone benchmark was published in the CACM (10/84 27:10 p1013) and converted by MANX from ADA to C. The Dhrystone benchmark was designed to produce a figure of merit for performance for systems software. For a full copy of the Manx Dhrystone and Whetstone benchmarks including timings for large memory models call Manx.

	Execution Time	Code Size	Compile/ Link Time
Sieve Benchmark			
Manx Aztec C86 2.2	11 secs	4,448	64 secs
Lattice 2.13	11 secs	21,902	98 secs
Mark Williams 2.0	12 secs	6,887	79 secs
Optimized C86 2.20G	13 secs	12,729	111 secs

Matrix Benchmark			
Manx Aztec C86 2.2	16 secs	7,804	92 secs
Lattice 2.13	29 secs	25,176	163 secs
Mark Williams 2.0	29 secs	10,847	107 secs
Optimized C86 2.20G	27 secs	13,766	134 secs

Dhrystone Benchmark			
Manx Aztec C86 2.2	36 secs	5,680	93 secs
Lattice 2.14	89 secs	20,404	117 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Optimized C86 2.20J	53 secs	11,009	172 secs

... with Less Effort

At Manx, we understand the value of development tools, specialized features, professional documentation and technical support in making it easier for you to produce quality software. That's why we bundle more than \$1000 worth of special features and tools with Manx Aztec C86-c, provide documentation that wins consistent praise for its clarity and completeness, and provide telephone access to experienced technical support for all Manx Aztec C products.

The following are some of the more important components of the Manx Aztec C86 Software Development System. Notice that many of the features that are bundled with Manx Aztec C86-c such as the debugger, Z editor, macro assembler, library source code, and ROM support are extra cost items with other systems.

Optimized C compiler	Symbolic Debugger
AS86 Macro Assembler	C Utility Library
LN86 Overlay Linker	DOS Function Library
Z (Vi) Source Editor-c	8087/80287 Sensing Lib
ROM Support Package-c	80186/80286 Support
Graphics Library	INTEL HEX Utility-c
CP/M-86 Library-c	Librarian
Screen Library	Graphics Library
Extensive UNIX Library	Object File Utilities
Library Source Code-c	Mixed memory models-c
Microsoft/Intel Object Option	Lattice, Microsoft, and C86 Interface
Small and Large memory models	Unitools (MAKE, DIFF and GREP)-c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c above are provided as special features of the Aztec C86-c system. Other items are provided with both.

Third party software can often greatly improve software quality or significantly reduce software development time. Manx works with third party software developers to assure the availability of a wide variety of high quality third party software for Aztec C. Third party software is available directly from Manx. Some of the better known software products available for Aztec C86 are:

HALO	C WINDOWS	Amber Windows
PHACT	Sunscreen	PLINK86
C-TREE	PANEL	FirstTime
PRE-C	Greentree	C Util Lib

Call Manx to order or for information on third party software.

... and Portable, Too!

Manx Aztec C is the most portable C development system available. Manx Aztec C is the only C development system available for all of the following: MS-DOS, PC-DOS, CP/M-86, Macintosh, CPM-80, Apple II, Radio Shack and Commodore. Manx Aztec C syntax and library routines conform as closely as possible to the letter and spirit of UNIX Version 7, System III and System V. Software created for one environment can be adapted to a host of others.

... Cross Compilation Saves Time and Resources

A comprehensive set of Cross compilers are available from Manx. Cross compilers allow development for a number of machines to be performed on a single faster host machine. Code is then downloaded and tested on the target machine. In some cases testing can also be done on the host machine. Using a PC or AT to produce and test ROM code is less expensive and more effective than specialized micro development systems.

To target development for PC-DOS, MS-DOS, CP/M-86, or ROM based 8088/8086/80186/80286 systems, Cross compilers are available from VAX/UNIX (\$2000), PDP-11/UNIX (\$1000), and the Apple Macintosh (\$750).

A wide variety of PC-DOS, MS-DOS, and CP/M-86, based cross compilers are available. The host system must be licensed for Aztec C86-c. The following targets are available for \$300 each:

Macintosh	Apple II	CP/M-80
TRS-80 III/IV	6502/6510/6511	8080/280

Call Manx for information on cross development to the 68000, 65816, Amiga, C64, C128, CP/M-68K, VRTX, and others.

For more information on XENIX, TOPVIEW and other specialized development, call Manx.

30 Day Guarantee

Any Manx Aztec C86-c or Manx Aztec C86-d development system can be returned within 30 days for a refund if it fails to meet your needs. Restrictions are that the original purchase must be directly from Manx, and shipped within the USA. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of Lattice C, C86, Mark Williams C, Wizard C, Digital Research C, and Microsoft C. Call Manx for details.

Aztec C Software Development Systems

To accommodate the widest possible audience of C users, Manx Aztec C is available in four configurations. Manx Aztec C86-c, Manx Aztec C86-d, Manx Aztec C86-p, and Manx Aztec C86-a.

Aztec C86-c (Commercial System) \$499

This is the best C software development system available for PC-DOS, MS-DOS, and CP/M-86. It is heavily bundled with time saving development tools and special features (see list). Initial purchase entitles the licensed user to one year of computer tracked updates. After one year, the update service can be renewed on an annual basis for a modest fee.

Aztec C86-d (Developer's System) \$299

This system includes all of the major elements of the Manx Aztec C86 Software Development System. Including Optimized C compiler, Macro Assembler, Linker, Librarian, Source Editor, and Symbolic Debugger. For price, performance, and professional features it is far superior to competing systems with list prices that are much higher. Aztec C86-d can be upgraded to Manx Aztec C86-c by paying the difference in list price plus \$10.

Aztec C86-p (Personal System) \$199

This system comes with a non-commercial license and can be upgraded.

Aztec C86-a (Apprentice System) \$99

This system is designed to assist in learning the C language.

How Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

There is no charge for standard domestic delivery. One day and two day express delivery is available to most areas for a minimal fee. Outside the USA standard delivery is \$20 additional and express delivery is sent freight collect. Manx takes care of export documents and the like. COD on export is available to some areas.

How To Get More Information

To get more information on Manx Aztec C and related products call 1-800-221-0440 or write to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized. Manx Aztec C is sold at some retail outlets under license to Manx.



To order or for information call:

800-221-0440

UNIX is a registered TM of AT&T Bell Laboratories, MS TM Microsoft, Inc., Halo TM Media Cybernetics, PANEL TM Roundhill Computer Systems Ltd., APPLE II and Macintosh TM Apple, Inc., TRS-80 TM Radio Shack, C64 TM Commodore Int., Plink 86, Pre-C TM Phoenix Software

Circle no. 62 on reader service card.

1390 Flynn Rd., Unit E
Camarillo, CA 93010
(805) 987-9781

Reader Service No. 131.

SoftLogic Solutions
530 Chestnut St.
Manchester, NH 03101
(800) 272-9900

Reader Service No. 133.

Phoenix Computer Products Corp.
1416 Providence Highway, Ste 220
Norwood, MA 02062
(617) 762-5030

Reader Service No. 135.

Morgan Computing Co., Inc.
P.O. Box 112730
Dallas, TX 75011
(214) 245-4763

Reader Service No. 137.

Genesis Microsystems Corp.
196 Castro St.
Mountain View, CA 94041
(415) 964-9001

Reader Service No. 139.

Level Five Research
4980 South A-1-A
Melbourne Beach, FL 32951
(305) 729-9046

Reader Service No. 141.

Human Edge Software Corp.
2445 Faber Place
Palo Alto, CA 94303
(415) 493-1593

Reader Service No. 143.

Poor Person Software
3721 Starr King Circle
Palo Alto, CA 94306
(415) 493-3735

Reader Service No. 145.

Minnow Bear Computers
P.O. Box 2233 Sta. A
Champaign, IL 61820-8233
(217) 398-6883

Reader Service No. 147.

Soft Advances
P.O. Box 49473
Austin, TX 78765
(512) 478-4763

Reader Service No. 149.

Selfware
3545 Chain Bridge Rd., Ste 3
Fairfax, VA 22030
(703) 352-2977

Reader Service No. 151.

Micro-Software Developers, Inc.
214½ West Main St.
St. Charles, IL 60174
(312) 377-5151

Reader Service No. 153.

Catalytix Corp.
55 Wheeler St.
Cambridge, MA 02138
(617) 497-2160

Reader Service No. 155.

Lattice C User's Group
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950

Reader Service No. 157.

Whitesmiths
97 Lowell Rd.
Concord, MA 01742
(617) 369-8499

Reader Service No. 159.

The Software Toolworks
15233 Ventura Blvd., Ste 1118
Sherman Oaks, CA 91403
(818) 986-4885

Reader Service No. 161.

Smith & Smith Associates
P.O. Box 160
Hunt Valley, MD 21030
(301) 666-8129

Reader Service No. 163.

Vance Info Systems
2818 Clay St.
San Francisco, CA 94115
(415) 922-6539

Reader Service No. 165.

Kriya
505 N. Lake Shore Dr. Suite 5510
Chicago, IL 60611
(312) 822-0624

Reader Service No. 167.

Dennis Brothers' MacTEP
Berkeley Mac Users Group
1442A Walnut St., Ste. 153
Berkeley, CA 94709

Reader Service No. 169.

Red Ryder
The FreeSoft Co.
10828 Lacklink
St. Louis, MO 63114
(314) 428-8057

Reader Service No. 171.

The IM Underground
715 Church St. #16
Ann Arbor, MI 48104

Reader Service No. 173.

MacTutor
P.O. Box 846
Placentia, CA 92670
(714) 993-9939

Reader Service No. 175.

The Mac Street Journal
P.O. Box 6686
Yorkville Station
New York, NY 10128

Reader Service No. 177.

MacBriefs
P.O. Box 2178
Huntington Beach, CA 92647
(714) 842-0518

Reader Service No. 179.

MicroSparc
45 Winthrop St.
Concord, MA 01742
(617) 371-1660

Reader Service No. 181.

InCider
CW Communications/Peterborough
Peterborough, NH 03458

Reader Service No. 183.

Pandora Software
P.O. Box 55
Clearfield, UT 84015

Reader Service No. 185.

IEEE
345 East 47th St.
New York, NY 10017

Reader Service No. 187.

Aymara Fab Lab
1974 Gardner St.
Sunnyvale, CA 94086

Reader Service No. 189.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service No. 198.

PERFORMANCE ACCELERATORS FOR CP/M-80 & MP/M-80

WSOPTION for WORDSTAR 3.0 & 3.3 (installs itself right into WordStar)

FAST

Speeds up action of WordStar functions so you spend less time waiting for WordStar to take action on your commands.

PRODUCTIVE

A superior print-while-edit capability is included. You can now use one terminal or micro to edit one file while printing another without reducing your typing speed!

CONVENIENT

At print time you can select 1 of 4 printers under CP/M or 1 of 8 printers under MP/M. Under MP/M you do a proper attach and detach of the printer so it is always free when you are not using it. You are informed if the printer is in use at print time so you do not hang up. Many additional features installed via menu.

MPMPLUS for MP/M-80

A group of programs and modules designed to improve your console and disk response under MP/M-80. A performance increase of 2 to 3 times is usual for disk I/O.

Send \$35.00 for each item plus \$2.00 postage.

INCLUDE DISK FORMAT REQUIRED

Continuum Microsystems Ltd. Use your Visa
21 McCarty Crescent or M.C.
Markham, Ontario
Canada L3P 4R4 (416) 294-8536
WordStar reg. MicroPro, CP/M MP/M reg. D.R.I.

Circle no. 29 on reader service card.

Quelo™ 68000 Software Development Tools

68000/68010 Assembler Package

Assembler, linker, object librarian and extensive indexed typeset manuals.

Conforms to Motorola structured assembler, publication M68KMASM[4]. Macros, cross reference and superb load map, 31 character symbols.

Optimized for CP/M-80, -86, -68K, MS-DOS, PC-DOS . \$ 595
Portable Source written in "C" . \$1495

Complete 68000 Development Package for MS-DOS

Lattice 68000 "C" Compiler and
Quelo 68000 Assembler Package . \$1095

68200 Assembler Package

Assembler and linker for Mostek MK68200.

Optimized for CP/M-80, MS-DOS, PC-DOS . \$ 595

For more information contact

Quelo Inc.
2464 33rd W. Suite #173
Seattle, WA 98199
Patrick Adams Phone (206) 285-2528
COD, Visa, MasterCard telex II (TWX) 910338171

CP/M, TM DRI, MS-DOS TM Microsoft, PC-DOS TM IBM.

Circle no. 61 on reader service card.

Now With Windowing! \$49.95 Basic Compiler

MTBASIC

Features:

- | | |
|---------------------------|-------------------------|
| Multitasking | Windowing |
| Handles interrupts | Interactive |
| Fast native code | Compiles quickly |
| Floating point | No runtime fee |

MTBASIC is a true native code compiler. It runs Bytes's Sept. '81 seive in 26 seconds; interpreters take over 1400 seconds! Because MTBASIC is multitasking, it can run up to 10 Basic routines at the same time, while displaying ten separate windows. Pop-up/down menus are a snap to implement.

MTBASIC combines the best of interpreters and compilers. To the programmer, MTBASIC appears to be an extremely fast interpreter. MTBASIC compiles a typical 100 line Basic program in 1 second, yet it generates blindingly fast code. No more waiting for long compilations.

AVAILABLE for CP/M (Z-80) and PC-DOS systems.

ORDERING: Specify format when ordering. We accept Visa, MC, checks and COD. Send \$49.95 plus \$3.50 shipping and handling (\$10 overseas) to:



P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

Circle no. 88 on reader service card.



COLOR BUSTER !!!



CONVERT THE RGB OUTPUT OF YOUR COLOR BOARD FOR THE IBM PC OR AN IBM PC COMPATIBLE TO COMPOSITE B/W WITH 16 LEVELS OF GRAY !

RUN ALL APPLICATIONS REQUIRING A COLOR MONITOR ON YOUR COMPOSITE BLACK AND WHITE MONOCHROME DISPLAY !

WITH THE ABILITY TO ELIMINATE OR HIGHLIGHT ANY COLOR GROUP, THE COLOR BUSTER OUTPERFORMS ANY OTHER COLOR TO B/W CONVERTER AVAILABLE.

BUY YOURS TODAY AT THIS SPECIAL INTRODUCTORY PRICE FOR A LIMITED TIME ONLY !

ONLY \$69.95

(Calif. residents add 6.5% tax)

STS ENTERPRISES
5469 ARLENE WAY
LIVERMORE, CA 94550
(415) 455-5358

Circle no. 47 on reader service card.

NOW C HERE! CROSS SOFTWARE for the NS32000

Also Available for IBM PC
INCLUDES:

- ★ Cross Assembler ★
- ★ Cross Linker ★
- ★ Debugger ★
- ★ N.S. ISE Support ★
- ★ Librarian ★
- ★ Pascal Cross Compiler ★
- ★ C Cross Compiler ★

U.S. prices start at \$500

SOLUTIONWARE

1283 Mt. View-Alviso Rd.
Suite B
Sunnyvale, Calif. 94089
408/745-7818 ★ TLX 4994264

Circle no. 118 on reader service card.

C Users' Group

Over 45 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group

415 E. Euclid • Box 97A
McPherson, KS 67460
(316) 241-1065

Circle no. 17 on reader service card.

ICs PROMPT DELIVERY!!! SAME DAY SHIPPING (USUALLY)

OUTSIDE OKLAHOMA: NO SALES TAX

8087-3 MATH \$110.00
8087-2 COPROCESSORS 162.00

DYNAMIC RAM

256K 256Kx1 120 ns \$ 5.65
256K 256Kx1 150 ns 4.45
64K 64Kx1 150 ns 1.10

EPROM

27C256 32Kx8 250 ns \$19.99
27256 32Kx8 250 ns 10.99
27128 16Kx8 250 ns 4.48
27C64 8Kx8 200 ns 7.85
2764 8Kx8 250 ns 2.95
2732A 4Kx8 250 ns 3.95

STATIC RAM

6264LP-15 8Kx8 150 ns \$6.57
6116LP-3 2Kx8 150 ns 2.24

OPEN 6 1/2 DAYS: WE CAN SHIP VIA FED-EX ON SAT.

MasterCard/VISA or UPS CASH COD

Factory New, Prime Parts

MICROPROCESSORS UNLIMITED

24,000 S. Peoria Ave. (918) 267-4961

BEGGS, OK 74421

Prices shown above are for May 20, 1985

Please call for current prices. Prices subject to change or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 6 PM CST can usually be delivered to you by the next morning. Via Federal Express Standard Air is \$6.00, or Priority One is \$11.00.

NO EXTRA COST ON F-EX SAT DELIVERY

QUANTITY ONE PRICES SHOWN

Circle no. 64 on reader service card.

FORTRAN PROGRAMMERS

Discover why you should be using F77L

the complete implementation of the ANSI FORTRAN 77 Standard for the IBM PC and compatibles.

If you are serious about your FORTRAN programming, you should be using F77L.

\$477



Lahey Computer Systems, Inc.

31244 Palos Verdes Drive West, Suite 243
Rancho Palos Verdes, California 90274
(213) 541-1200

Serving the FORTRAN community since 1969

Circle no. 104 on reader service card.

NEW! Advanced Trace86™

Symbolic Debugger & Assembler Combo

- Full-screen trace with single stepping; Even backstepping!
- Write & Edit COM & EXE programs
- Conditional breakpoints (programmable)
- Switch between trace and output screen; Or set up two monitors
- 8087, 80186, 80286, 80287 support
- Write labels & comments on code
- Polish hex/decimal calculator
- and more ... Priced at \$175.00

To order or request more information contact:

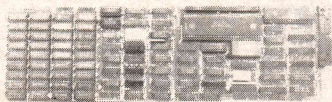


Morgan Computing Co., Inc.

2520 Tarpley Rd. Suite 500
(214) 245-4763 Carrollton, TX 75006

Circle no. 128 on reader service card.

68000 CO-PROCESSING For IBM PC, PC/XT and COMPATIBLE SYSTEMS



Now you can add the MOTOROLA 68000 16/32 Bit Processor to your PC via use of the Pro 68 Advanced Technology Co-Processor. Enjoy all of the performance benefits of the 68000 processor without sacrificing your current PC system. Consider these impressive standard features of Pro 68:

- High Speed MOTOROLA 68000 micro processor
- 10Mhz no wait state design (3 times faster than the IBM PC/AT)
- True 16/32 bit technology
- For use on IBM PC, PC/XT or compatible systems
- On board 16 bit parity checked memory, 256K to 1024K
- Two serial I/O ports for multi user interface
- Provisions for the high speed NS32081 math processor
- High speed proprietary dual port host bus interface
- Parallel or array processing via multi processor architecture
- MS/PC DOS RAM disk driver program
- Choice of two popular integrated 16/32 bit operating systems:
 - CPM68K from Digital Research Inc.
 - Full suite of development tools
 - "C" compiler with floats and UNIX I/O library
 - Many third party compatible languages and applications
- OS9/6800 from MICROWARE Corporation
 - UNIX look alike with multi user / multi tasking, shell, hierarchical disk directory, record and file lock, pipes and filters
 - Full suite of development tools
 - UNIX V compatible "C" compiler
 - Optional languages include BASIC, ISO PASCAL, FORTRAN 77.

Pricing from \$1195 includes Pro68 with 256K, OS, and MS/PC DOS RAM disk driver. HSC also manufactures and markets a full line of co-processors and RAM disks for use on Z80 based systems.

DISTRIBUTORS:

Australia-Computer Transition Systems
...03-537-2768
Great Britain-System Science
...01-248-062
West Germany-DSC International
...089-723-1125
Canada Remote Systems
...416-239-2835

Dealer, Distributor and OEM inquiries invited.



Hallock Systems Co., Inc.
267 North Main Street
Herkimer, NY 13350
(315) 866-7125

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.
4	AGSComputers, Inc.	C-2	41	MicroSmith	55
*	Alcor/MIX Software	1	64	Microprocessors Unlimited	127
1	Alpha Computer Service	67	66	Mitec	103
16	Arity Corporation	23	128	Morgan Computing	127
5	Ashton Tate	30-31	42	Multi-Tech Electronics	93
130	Automata Design Associates	61	30	Mystic Canyon Software	106
12	BD Software, Inc.	51	25	Ordinate Solutions	83
8	Blaise Computing	C-3	60	Paul Mann & Associates	100
14	Borland International	C-4	76	Personal Tex	65
17	C User's Group	127	70	Phoenix Computer Products	13
18	C Ware	61	69	Plu Perfect Systems	69
21	Chalcedony	10	71	Poor Persons Software	67
38	Command Software Systems	89	67	Programmer's Connection	49
22	Computer Helper Industries, Inc.	61	72	Programmer's Shop	35
96	Computer Innovations	121	61	Quelo	127
57	Consulair Corp.	91	79	Rational Systems, Inc.	73
29	Continuum Microsystems Ltd.	127	49	Relational Database Systems	45
27	Creative Solutions	43	80	Revasco	91
28	D & W Digital	117	78	SLR Systems	73
*	DDJ Classified Advertising	115	47	STS Enterprises	127
31	Data Base Decisions	119	85	SemiDisk Systems	67
3	Data Technology	2	86	Shaw American Technologies	69
2	Davong	7	52	Signum Systems	55
33	Digital Research Computers	15	63	Soft Advances	117
35	Ecosoft, Inc.	117	88	Softaid, Inc.	127
13	Edward K. Ream	100	*	Softfocus	103
36	Essential Software	17	90	Software Horizons, Inc.	123
37	Fair-com	65	92	Softway, Inc.	19
40	Fox Software, Inc.	47	75	Solution Systems	71
*	Gimpel Software	54	93	Solution Systems	71
*	Gimpel Software	111	94	Solution Systems	71
43	Greenleaf Software, Inc.	22	95	Solution Systems	71
26	Hallock Systems Consultants	127	102	Solution Systems	39
44	Harvard Softworks	65	118	Solutionware	127
46	Illyes Systems	18	59	Speedware	83
*	Integral Quality	124	65	Spruce Technologies Corp.	3
23	Integral Systems	55	50	Systems Guild	109
15	Integrand Research Corp.	107	34	TSF	107
*	J. D. Owens & Assoc.	24	39	Tiny Tek Inc.	119
*	JDR Microdevices	53	81	Turbo Power Software	85
55	Laboratory Microsystems Inc.	85	91	US Software	73
104	Lahey Computer Systems	127	77	UniPress Software	9
58	Lattice, Inc.	107	112	Wendin, Inc.	11
9	Logique	69	116	Wizard Systems	103
24	Lugaru	83	*	DDJ Back Issues	122
62	Manx Software	125	*	DDJ Bound Volume	113
84	Megamax, Inc.	91	*	DDJ C Compiler	85
53	Micro Motion	77	*	DDJ Subscription Problem	77
134	Micro Software Developers	77	*	DDJ Subscription	97

Advertising Sales Offices

East Coast
Walter Andrzejewski (617) 567-8361

Midwest/West Central
Michele Beaty (317) 875-0557

Northern California/Northwest
Lisa Boudreau (415) 424-0600

Southern California/Southwest
Beth Dudas (714) 643-9439

Classified Advertising
Tim Ortiz (415) 424-0600

Advertising Director
Stephen Friedman (415) 424-0600

Advertising Coordinators
Jay Horvath (415) 424-0600
Alice Abrams (415) 424-0600

PERFORMANCE PACKAGE

Blaise Computing Inc. introduces the PERFORMANCE PACKAGE™ for Turbo Pascal programmers.

TURBO PASCAL

Turbo ASYNCH™ With Turbo ASYNCH, you can be in constant touch with the world without ever leaving the console. Rapid transit at its best. Turbo ASYNCH is designed to let you incorporate asynchronous communication capabilities into your Turbo Pascal application programs, and it will drive any asynchronous device via the RS232 ports, like printers, plotters, modems or even other computers. Turbo ASYNCH is fast, accurate and lives up to its specs. Features include...

- ◆ Initialization of the COM ports allowing you to set all transmission options.
- ◆ Interrupt processing.
- ◆ Data transfer between circular queues and communications ports.
- ◆ Simultaneous buffered input and output to both COM ports.
- ◆ Transmission speeds up to 9600 Baud.
- ◆ Input and output queues as large as you wish.
- ◆ XON/XOFF protocol.

The underlying functions of Turbo ASYNCH are carefully crafted in assembler for efficiency, and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The interface to the assembler routines is written in Turbo Pascal.

The Turbo Pascal PERFORMANCE PACKAGE™ is for the serious Turbo Pascal programmer who wants quality tools to develop applications. Every system comes with a comprehensive User Reference Manual, all source code and useful sample programs. They require an IBM PC or compatible, utilizing MS-DOS version 2.0 or later. There are no royalties for incorporating PERFORMANCE PACKAGE functions into your applications.

Turbo POWER TOOLS and Turbo ASYNCH sell for \$99.95 each, and they may be ordered directly from Blaise Computing Inc. To order, call (415) 540-5441.

◆
BLAISE COMPUTING INC.

BLAISE

NEW!

Turbo POWER TOOLS™

Turbo POWER TOOLS is a sleek new series of procedures designed specifically to complement Turbo Pascal on IBM and compatible computers. Every component in Turbo POWER TOOLS is precision engineered to give you fluid and responsive handling, with all the options you need packed into its clean lines. High performance and full instrumentation, including...

- ◆ Extensive string handling to complement the powerful Turbo Pascal functions.
- ◆ Screen support and window management, giving you fast direct access to the screen without using BIOS calls.
- ◆ Access to BIOS and DOS services, including DOS 3.0 and the IBM AT.
- ◆ Full program control by allowing you to execute any other program from within your Turbo Pascal application.
- ◆ Interrupt service routines written entirely in Turbo Pascal. Assembly code is not required even to service hardware interrupts like the keyboard or clock.

Using Turbo POWER TOOLS, you can now "filter" the keyboard or even DOS, and create your own "sidekickable" applications.

YES, send me the Best for the Best! Enclosed is _____ for
☐ Turbo ASYNCH ☐ Turbo POWER TOOLS. (CA residents add
6 1/2% Sales Tax. All orders add \$6.00 for shipping.)

Name: _____ Phone: _____

Shipping Address: _____ State: _____ Zip: _____

City: _____ Exp. Date: _____

VISA or MC #: _____

Turbo Pascal is a trademark of Borland International. Turbo POWER TOOLS, Turbo ASYNCH and PERFORMANCE PACKAGE are trademarks of Blaise Computing Inc. IBM is a registered trademark of International Business Machines Corporation. MS-DOS is a trademark of Microsoft Corporation.

Watch us!

- ◆ 2034 BLAKE STREET
- ◆ BERKELEY, CA 94704
- ◆ (415) 540-5441

They said it couldn't be done. Borland Did It. Turbo Pascal 3.0

**MOST SIGNIFICANT PRODUCT
OF THE YEAR - PC WEEK**

The industry standard

With more than 250,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use!

COMPILATION SPEED	8.1	16	206
EXECUTION SPEED	9 ^{SEC}	13 ^{SEC}	20 ^{SEC}
CODE SIZE	12 K	12 K	35 K
BUILT-IN INTERACTIVE EDITOR	YES	YES	NO
ONE STEP COMPILE (NO LINKING NECESSARY)	YES	YES	NO
COMPILER SIZE	39K	35K	300K+
TURTLE GRAPHICS	YES	NO	YES
BCD OPTION	YES	\$54 ⁹⁵	\$295 ⁰⁰
PRICE	\$69 ⁹⁵		

TURBO 3.0 **TURBO 2.0** **MS PASCAL**

The best just got better: Introducing Turbo Pascal 3.0

We just added a whole range of exciting new features to Turbo Pascal:

- First, the world's fastest Pascal compiler just got faster. Turbo Pascal 3.0 (16 bit version) compiles twice as fast as Turbo Pascal 2.0! No kidding.
- Then, we totally rewrote the file I/O system, and we also now support I/O redirection.
- For the IBM PC versions, we've even added "turtle graphics" and full tree directory support.
- For all 16 Bit versions, we now offer two additional options: 8087 math coprocessor support for intensive calculations and Binary Coded Decimals (BCD) for business applications.
- And much much more.

The Critics' Choice.

Jeff Duntemann, PC Magazine: "Language deal of the century . . . Turbo Pascal: It introduces a new programming environment and runs like magic."

Dave Garland, Popular Computing: "Most Pascal compilers barely fit on a disk, but Turbo Pascal packs an editor, compiler, linker, and runtime library into just 39K bytes of random-access memory."

Jerry Pournelle, BYTE: "What I think the computer industry is headed for: well documented, standard, plenty of good features, and a reasonable price."

Portability.

Turbo Pascal is available today for most computers running PC DOS, MS DOS, CP/M 80 or CP/M 86. A XENIX version of Turbo Pascal will soon be announced, and before the end of the year, Turbo Pascal will be running on most 68000 based microcomputers.

An Offer You Can't Refuse.

Until June 1st, 1985, you can get Turbo Pascal 3.0 for only \$69.95. Turbo Pascal 3.0, equipped with either the BCD or 8087 options, is available for an additional \$39.95 or Turbo Pascal 3.0 with both options for only \$124.95. As a matter of fact, if you own a 16-Bit computer and are serious about programming, you might as well get both options right away and save almost \$25.

Update policy.

As always, our first commitment is to our customers. You built Borland and we will always honor your support.

So, to make your upgrade to the exciting new version of Turbo Pascal 3.0 easy, we will accept your original Turbo Pascal disk (in a bend-proof container) for a trade-in credit of \$39.95 and your Turbo87 original disk for \$59.95. This trade-in credit may only be applied toward the purchase of Turbo Pascal 3.0 and its additional BCD and 8087 options (trade-in offer is only valid directly through Borland and until June 1st, 1985).

(*) Benchmark run on an IBM PC using MS Pascal version 3.2 and the DOS linker version 2.6. The 179 line program used is the "Gauss-Seidel" program out of Alan R. Miller's book: *Pascal programs for scientists and engineers* (Sybex, page 128) with a 3 dimensional non-singular matrix and a relaxation coefficient of 1.0.

TURBO PASCAL

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

Carefully Describe your Computer System!

Mine is: ☐ 8 bit ☐ 16 bit

I Use: ☐ PC-DOS ☐ MS-DOS

☐ CP/M 80 ☐ CP/M 86

My computer's name/model is: _____

The disk size I use is:

☐ 3 1/2" ☐ 5 1/4" ☐ 8"

Name: _____

Shipping Address: _____

City: _____ Zip: _____

State: _____

Telephone: _____

YES! I want the Best! Quantity

Please send:

Pascal 3.0 \$ 69.95 _____

Pascal w/8087 \$109.90 _____

Pascal w/BCD \$109.90 _____

Pascal w/8087 & BCD \$124.95 (SAVE \$24.90) _____

*These prices include shipping to all U.S. cities. All foreign orders add \$10 per product ordered.

Subtotal (CA 6% tax) _____

Trade-in Credit Claimed: _____

Amount Enclosed: _____

Payment: ☐ VISA ☐ MC ☐ BankDraft ☐ Check

Credit Card Expir. Date: _____

Card #: _____

For update: original Turbo disk must accompany order

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax. Outside USA: add \$10 and make payment by bank draft, payable in US dollars drawn on a US bank.

BORLAND
INTERNATIONAL

Software's Newest Direction
4585 Scotts Valley Drive
Scotts Valley, CA 95066
TELEX 172373

Turbo Pascal is a registered trademark of Borland International, Inc.
PC Week is a trademark of Ziff-Davis Pub. Co.
Circle no. 14 on reader service card.